UNIVERSIDAD
DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL

Departamento

Área de conocimiento

PROYECTO FIN DE CARRERA

Sistema didáctico para programación de robots móviles en Matlab/Simulink

Autor:

Juan Enrique Gil Lozano

Director:

Jesús Manuel Gómez de Gabriel

Titulación:

Ingeniero en Automática y Electrónica Industrial

MÁLAGA, julio de 2014

A mis padres.

Si una persona es perseverante, aunque sea dura de entendimiento, se hará inteligente; y aunque sea débil se transformará en fuerte.

Leonardo Da Vinci

Índice general

I MEMORIA **11**

1. Introducción **12**

1.1. Objetivos 13

1.2. Requerimientos del Proyecto 13

1.3. Organización de la Memoria 14

2. Antecedentes **15**

2.1. Lego Mindstorms NXT 15

2.2. Arduino Robot 16

3. Plataforma Robótica **17**

3.1. Componentes Principales 17

3.1.1. Plataforma Base 17

3.1.2. Arduino Mega 18

3.1.3. Motor, Ruedas y Soportes 22

3.1.4. Controlador de Motores 23

3.1.5. Sensor de Distancias 24

3.1.6. Conector Bluetooth 25

3.1.7. Bateria 25

3.2. Distribución y Montaje 26

4. PCB Shield de Conexiones **31**

4.1. Requisitos 32

4.2. Diseño de la PCB 32

4.2.1. Cálculo del Ancho de las Pistas 34

4.3. Fabricación de la Shield 36

5. Drivers de Arduino para Simulink **39**

5.1. ¿Qué es un Driver? 39

5.2. Instalación del Soporte Arduino en Simulink 39
5.3. Arduino en Simulink 41
5.4. Guía de Creación de Drivers 42
5.5. Bloques Desarrollados en el Proyecto 46
5.5.1. Bloque DualQuadEncoder 46
5.5.2. Bloque Sensor de Distancia 53

6. Prácticas **55**
6.1. Práctica 1: Control de Posición y Velocidad (PID) 55
6.1.1. Objetivos de la Práctica 55
6.1.2. Requisitos 56
6.1.3. Esquema de Conexiones 56
6.1.4. Guía de Realización 57
6.2. Práctica 2: Sistema de Navegación Reactivo 65
6.2.1. Objetivos de la Práctica 65
6.2.2. Requisitos 65
6.2.3. Esquema de Conexiones 66
6.2.4. Guía de Realización 66

7. Conclusiones y Trabajos Futuros **71**
7.1. Trabajos Futuros 72

Bibliografía **72**

II PRESUPUESTO **75**

1. Plataforma Móvil **76**

III ANEXOS **77**

A - Especificaciones para los Motores EMG30 y para los Soportes **78**

B - HC-SR04 Manual de Usuario **81**

C - HC-06 Manual de Usuario **92**

D - Esquemático del Arduino Mega 2560 **109**

E - Esquema de la Plataforma Base **111**

IV PLANOS Y ESQUEMAS 113

1. Esquemático PieroShield **114**

2. Diseño PieroShield **115**

Índice de figuras

1.0.1.Sistemas hardware compatibles con Simulink 12

2.1.1.Lego Mindstorms NXT 15
2.2.1.Arduino Robot 16

3.1.1.Modelo 3d de la plataforma inicial 17
3.1.2.Modelo 3d de la plataforma evolucionada 18
3.1.3.Arduino Mega 19
3.1.4.Motor EMG30 22
3.1.5.Rueda de goma 22
3.1.6.Soporte para el motor 23
3.1.7.L298N 23
3.1.8.HC-SR04 24
3.1.9.Módulo bluetooth 25
3.1.10Batería LIPO de 3 celdas 26
3.2.1.Localización del L298N 26
3.2.2.Abrazaderas sobre la bateria 27
3.2.3.Montaje de la batería sobre la plataforma 27
3.2.4.Esquema de conexión 28
3.2.5.Esuqema de conexión de los motores 28
3.2.6.Situación Arduino Mega 29
3.2.7.Soporte conector bluetooth 29
3.2.8.Soporte conector bluetooth 30

4.0.1.PCB 31
4.2.1.Diseño Final 33
4.3.1.PCB Final 37
4.3.2.Conectores Utilizados 38
4.3.3.Shield sobre Arduino 38

5.1.1.Driver 39
5.2.1.Fuente del Paquete 40
5.2.2.Listado de Componentes 40
5.2.3.Login Cuenta de Mathwork 41
5.4.1.Data Properties 43
5.4.2.Generación del Fichero TLC 43
5.4.3.Panel de Inicialización 44
5.4.4.Código de Inicialización 44
5.4.5.Outputs update 45
5.4.6.Librerías 45
5.5.1.Señales encoders 47
5.5.2.Bloque Simulink Encoder 48
5.5.3.Parámetros Bloque Encoder 48
5.5.4.Registros TCCR 51
5.5.5.Bloque Sensor de Distancia 53
5.5.6.Librería «NewPing» 53
5.5.7.Parámetros Sensor Ultrasónico 54

6.1.1.Esquema de Conexiones 56
6.1.2.Esquema Básico Motores 57
6.1.3.Esquema Simulink PID posición 58
6.1.4.Preferencias PID 59
6.1.5.Preferencias PID Avanzado 59
6.1.6.Respuesta PID Posición 61
6.1.7.PID posición Apartado 2 62
6.1.8.Resultado Apartado 2 62
6.1.9.Esquema Control de Velocidad 63
6.1.10.PID Velocidad 64
6.2.1.Esquema de Conexiones 66
6.2.2.Diagrama de Estados 67
6.2.3.Diagrama de Bloques Simulink 67
6.2.4.DataStoreMemory 68
6.2.5.Bloque «send msg» 69

Índice de tablas

3.1. Resumen de las Características de Arduino Mega 19
3.2. Características L298N 24

4.1. Características de la PCB 36
4.2. Especificaciones de Manufactura 36

5.1. Bloques Librería Arduino de Simulink 41
5.2. Descripción del CS (Clock Select Bit) 51

6.1. Conexiones de los Motores 57
6.2. Regla de Ziegler-Nichols 60
6.3. Parámetros PID 61
6.4. Ganancias Control de Velocidad 63

I

MEMORIA

Capítulo 1

Introducción

El aprendizaje basado aplicado es un método de enseñanza y aprendizaje que implica a los estudiantes en la resolución de problemas de ingeniería reales. Para que este método sea eficaz, el profesor debe identificar un problema abordable, pero con una dificultad suficiente para estimular el proceso de aprendizaje.

El uso de elementos físicos plantea al alumno una versión completa del problema a resolver, en lugar de un modelo virtual simplificado, a la vez que aumenta su motivación. Desafortunadamente, muchos sistemas de desarrollo hardware son complejos y distraen al alumno con problemas hardware de bajo nivel.

Las herramientas de gran reconocimiento académico como son Matlab y Simulink permiten el desarrollo de los modelos teóricos y su simulación, además hoy día permiten su integración con las plataformas de desarrollo hardware de bajo coste que más popularidad han adquirido recientemente, como los sistemas Arduino, Lego NXT, Beaglebone y Raspberry Pi, entre otras.

Figura 1.0.1: Sistemas hardware compatibles con Simulink

La principal ventaja de este enfoque consiste en la integración de los robots de bajo coste con una herramienta software académica con la que los alumnos ya se encuentran

familiarizados, sin necesidad de utilizar sistemas adicionales, ni convertir o traducir el código obtenido.

Esto debería permitir la resolución de problemas con sistemas físicos en el mismo tiempo que una práctica con simulación. Asimismo, el hecho de no necesitar nuevas herramientas debería minimizar el tiempo invertido por el profesor y por el alumno en el aprendizaje del sistema físico.

1.1. Objetivos

El objetivo principal del presente proyecto consiste en desarrollar una plataforma móvil y un conjunto de prácticas que serán empleadas en diversas asignaturas relacionadas con la robótica y que incluyen prácticas de modelado de robots.

Para ello, se construirá una plataforma robótica móvil de bajo coste que contará con diversos sensores y actuadores. Además, el corazón de este robot estará formado por el sistema micocontrolador compatible con Simulink. Dado que el proyecto esta enfocado a temas docentes, se deberá construir la plataforma de manera se facilite la conexión con los diferentes sensores y actuadores. Así como el la interconexión con otros dispositivos utlizando el ampliamente extendido protocolo de comunicaciones I2C.

Una característica importante que diferencia este robot de otras plataformas de hobby, es que debe contar con motores con encoders para poder realizar un control de velocidad, seguir trayectorias y obtener información odométrica.

La programación de este conjunto de prácticas se realizará utilizando las herramientas mencionadas anteriormente, Matlab y Simulink. Para ello, se construirá una librería en Simulink que permitirá al alumno utilizar cada uno de los sensores y actuadores del sistema.

1.2. Requerimientos del Proyecto

Una vez conocidos los objetivos del proyecto, en este apartado se van a listar y describir los requerimientos que se deben seguir en el desarrollo del proyecto para poder alcanzar la consecución de dichos objetivos.

- Se debe diseñar una plataforma robótica móvil de bajo coste.
- Se deben diseñar prácticas relacionadas con los aspectos fundamentales de la robótica.
- Las prácticas de la plataforma deberán llevase a cabo utilizando Simulink.

- La plataforma deberá reducir el tiempo empleado tanto por los profesores como por los alumnos en los aspectos técnicos, como el montaje, conexionado, programación... para poder centrarse en los temas realmente relaciados con la robótica.
- El núcleo de la plataforma deberá estar compuesto por un sistema hardware de bajo coste compatible con Simulink como Arduino o Raspberry PI.
- La plataforma móvil debe incluir motores con encoders para poder permitir utilizar técnicas de control sobre ellos.
- La plataforma debe disponer de un sistema de conexiones que facilite la conexión de multiple sensores, para ello se debe diseñar una placa de conexiones compatible con el tipo de sistema empleado (Arduino, Raspberry Pi) que disponga de conectores tanto analógicos como digitales compatibles con los diferentes tipos de sensores.

1.3. Organización de la Memoria

La presente memoria esta estructurada en 7 capítulos que a su vez se dividen en varias secciones y subsecciones.

- En el primer capítulo se presenta a modo introductorio el proyecto, se definen los principales objetivos que se deben llevar a cabo y los requisitos del proyecto.
- En el segundo capítulo se desbriben los antecedentes del proyecto.
- En el tercer capítulo se describen los diferentes componentes que se han elegido para desarrollar la plataforma y además se describe su proceso de montaje.
- El cuarto capítulo se centra en la PCB shield de conexiones diseñada y desarrollada para el proyecto.
- En el capítulo quinto se describe el soporte para Arduino en Simulink, además se describen los diferentes drivers creados especificamente para este proyecto.
- El capítulo sexto se centra en las prácticas, en la primera sección se describe la práctica de control de posición y velocidad. Y la segunda sección la práctica de sistema de navegación basada en diagrama de estados.
- En el séptimo y último capítulo se describen las conclusiones del proyecto y los posibles trabajos futuros.

Capítulo 2

Antecedentes

En este capítulo se van a describir las diferentes plataformas móviles similares en cuanto a funcionalidades a la que se va a desarrollar en este proyecto, como Lego Mindstrom NXT o arduino robot.

2.1. Lego Mindstorms NXT

Lego Mindstorms NXT es un juego de robótica fabricado por la empresa Lego, el cual posee elementos básicos de las teoŕıas roboticas, como la uníon de piezas y la programación de acciones, en forma interactiva. Este robot fue comercializado por primera vez en septiembre de 1998. A parte de como juego también se vende como herramienta educacional, lo que originalmente se penśo en una sociedad entre Lego y el MIT finalmente acabo con Lego financiando la investigación y pudiendo usar las aplicaciones creadas sin pagar costes al MIT.

Figura 2.1.1: Lego Mindstorms NXT

El paquete básico incorpora disversos sensores, servomotores y el procesador. Resulta bastance fácil realizar pequeños montajes robóticos gracias a los componentes Lego y además Mathwork ofrece soporte para poder ser utilizado sobre Simulink de manera fácil. El problema de este sistema es el precio, el paquete incial con todos estos sensores tiene un precio superior a los 380 €. Por lo tanto no cumple uno de los requerimientos principales del presente proyecto consistente en llevar a cabo una plataforma robótica de bajo coste.

2.2. Arduino Robot

Arduino Robot se trata de un robot que consta de dos placas circulares que se apoyan sobre dos ruedas para aportar movilidad a este dispositivo. Ha sido diseñado en colaboración con Complubot, la Asociación de Robótica Educativa con sede en Madrid que ha logrado proclamarse cuatro veces ganadora de la competición de fútbol robótico Robocup Junior. El resultado del proyecto llamado Lottie Lemon es un sistema perfecto para iniciarse en el mundo de la robótica.

Figura 2.2.1: Arduino Robot

Por un lado tenemos la placa de control, que recibe la respuesta de los sensores y decide cómo operar, y por otro la placa motora, que controla el motor que imprime el movimiento a las ruedas. Cada placa es un sistema Arduino completo en sí misma, y ambas se pueden programar usando el IDE de Arduino. Sin embargo, uno de los problemas de este robot al igual que ocurre con el Lego Mindstorm es el alto precio, 300 $, careciendo de sensores y conectividad bluetooth.

Capítulo 3

Plataforma Robótica

En este capítulo se van a describir los componentes que se han elegido para desarrollar la plataforma y también se especificará la distribución de los diferentes componentes sobre la plataforma.

3.1. Componentes Principales

3.1.1. Plataforma Base

La platarforma base está construida en metacrilato de 5mm. Cuenta con las perforaciones necesarias para montar sobre ella los soportes de los motores (se describirán en la subsección 3.1.3) en la parte trasera, una rueda loca pasiva en la parte delantera, el Arduino UNO en la parte central y los soportes para sensores en la parte delantera. El esquema de esta plataforma se ha incluido en el Anexo E. En la figura 3.1.1 se observa un modelo 3d de la plataforma, donde se pueden observar la distribución de los componentes previamente descritos.

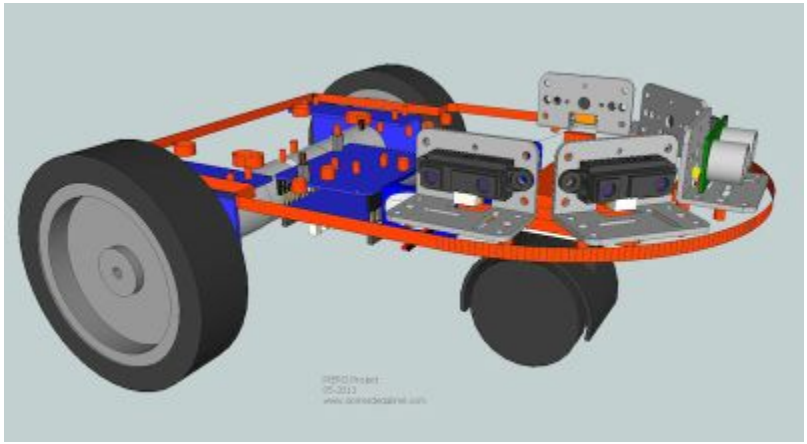


Figura 3.1.1: Modelo 3d de la plataforma inicial

Además se ha desarrollado en paralelo a este trabajo una evolución de esta plataforma, que dispone de una planta en forma circular con un diámetro de 30cm. El trabajo realizado en este proyecto es totalmente compatible con esta nueva plataforma, dado que se utilizan los mismos componentes, a pesar de que la forma haya evolucionado. En la siguiente figura se observa esta nueva plataforma.

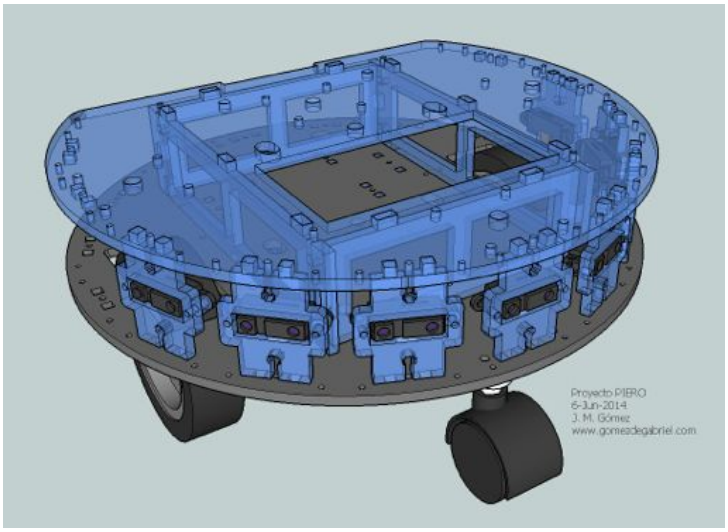


Figura 3.1.2: Modelo 3d de la plataforma evolucionada

Los planos y esquemas 3d de todas las versiones de la plataforma están disponibles para su descarga en la web del proyecto PIERO [4].

3.1.2. Arduino Mega

Aunque inicialmente existían varias alternativas como Arduino Uno, Arduino Mega o Raspberry Pi, finalmente se ha elegido Arduino Mega debido a diversas razones. A pesar de que el precio es poco superior al Arduino Uno, Arduino Mega dispone de mayor número de interrupciones reales, mayor número de salidas analógicas, mayor número de salidas digitales y a parte del apartado hardware también destaca en que Simulink ofrece servicios de depuración en tiempo real, sin embargo no lo ofrece para Arduino Uno. En cuanto la Raspberry Pi, se descartó frente a Arduino Mega puesto que el precio es superior y también ofrece menos compatibilidad con diversos paquetes de Simulink como por ejemplo la librería «StateFlow».

Figura 3.1.3: Arduino Mega

El Arduino Mega 2560 es una placa microcontrolador basada ATmega2560 (se adjunta el esquemático del Arduino Mega 2560 en el Anexo D). Tiene 54 entradas/salidas digitales (de las cuales 15 proporcionan salida PWM), 16 entradas digitales, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz, conexión USB, entrada de corriente, conector ICSP y botón de reset. Contiene todo lo necesario para hacer funcionar el microcontrolador; simplemente conectálo al ordenador con el cable USB o aliméntalo con un trasformador o batería para empezar. El Mega es compatible con la mayoría de shields diseñados para el Arduino Duemilanove o Diecimila. A continuación se muestra una tabla resumen con sus características:

Microcontrolador	ATmega2560
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines E/S digitales	54 (15 proporcionan salida PWM)
Pines de entrada analógica	16
Intensidad por pin	40 mA
Intensidad en pin 3.3V	50 mA
Memoria Flash	256 KB de las cuales 8 KB las usa el gestor de arranque(bootloader)
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

Tabla 3.1: Resumen de las Características de Arduino Mega

El Arduino Mega puede ser alimentado vía la conexión USB o con una fuente de alimentación externa. El origen de la alimentación se selecciona automáticamente. Las fuentes de alimentación externas (no-USB) pueden ser tanto un transformador o una batería. El transformador se puede conectar usando un conector macho de 2.1mm con centro positivo en el

conector hembra de la placa. Los cables de la batería puede conectarse a los pines Gnd y Vin en los conectores de alimentación (POWER) La placa puede trabajar con una alimentación externa de entre 6 a 20 voltios. Si el voltaje suministrado es inferior a 7V el pin de 5V puede proporcionar menos de 5 Voltios y la placa puede volverse inestable, si se usan mas de 12V los reguladores de voltaje se pueden sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios. Los pines de alimentación son los siguientes:

VIN. La entrada de voltaje a la placa Arduino cando se esta usando una fuente externa de alimentación (en opuesto a los 5 voltios de la conexión USB). Se puede proporcionar voltaje a través de este pin, o, si se esta alimentado a través de la conexión de 2.1mm , acceder a ella a través de est pin.

5V. La fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador integrado en la placa, o proporcionada directamente por el USB o otra fuente estabilizada de 5V.

3V3. Una fuente de voltaje a 3.3 voltios generada en el chip FTDI integrado en la placa. La corriente máxima soportada 50mA.

GND. Pines de toma de tierra.

El ATmega2560 tiene 256KB de memoria flash para almacenar código (8KB son usados para el arranque del sistema(bootloader).El ATmega1280 tiene 8 KB de memoria SRAM y 4KB de EEPROM.

Cada uno de los 54 pines digitales en el Arduino pueden utilizarse como entradas o como salidas usando las funciones «pinMode()», «digitalWrite()», y «digitalRead()». Las E/S operan a 5 voltios. Cada pin puede proporcionar o recibir una intensidad maxima de 40mA y tiene una resistencia interna (desconectada por defecto) de 20-50kOhms. Además, algunos pines tienen funciones especializadas:

- Serie: 0 (RX) y 1 (TX),Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX). Usado para recibir (RX) transmitir (TX) datos a través de puerto serie TTL. Los pines Serie: 0 (RX) y 1 (TX) estan conectados a los pines correspondientes del chip FTDI USB-to-TTL.

- Interrupciones Externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3), y 21 (interrupción 2).. Estos pines se pueden configurar para lanzar una interrupción en un valor LOW(0V), en flancos de subida o bajada (cambio de LOW a HIGH(5V) o viceversa), o en cambios de valor. Ver la función attachInterrupt() para as detalles.

- PWM: de 2 a 13 y 44 a 46. Proporciona una salida PWM (Pulse Wave Modulation, modulación de onda por pulsos) de 8 bits de resolución (valores de 0 a 255) a traves de la función «analogWrite()».

- SPI: 50 (SS), 51 (MOSI), 52 (MISO), 53 (SCK). Estos pines proporcionan comunicación SPI, que a pesar de que el hardware la proporcione actualmente no esta incluido en el lenguaje Arduino.

- LED: 13. Hay un LED integrado en la placa conectado al pin digital 13, cuando este pin tiene un valor HIGH(5V) el LED se enciende y cuando este tiene un valor LOW(0V) este se apaga.

El Mega tiene 16 entradas analógicas, y cada una de ellas proporciona una resolución de 10bits (1024 valores). Por defecto se mide de tierra a 5 voltios, aunque es posible cambiar la cota superior de este rango usando el pin AREF y la función analogReference(). Además algunos pines tienen funciones especializadas:

- I2C: 20 (SDA) y 21 (SCL). Soporte del protocolo de comunicaciones I2C (TWI) usando la librería Wire.

Hay unos otros pines en la placa:

- AREF. Voltaje de referencia para la entradas analogicas. Usado por «analogReference()».

- Reset. Suministrar un valor LOW(0V) para reiniciar el microcontrolador. Típicamente usado para añadir un botón de reset a los shields que no dejan acceso a este botón en la placa.

EL Arduino Mega facilita en varios aspectos la comunicación con el ordenador, otro Arduino o otros microcontroladores. El ATmega2560 proporciona cuatro puertos de comunicación vía serie UART TTL (5V).

El Arduino Mega tiene un multifusible reinicializable que protege la conexión USB de tu ordenador de cortocircuitos y sobretensiones. A aparte que la mayoría de ordenadores proporcionan su propia protección interna, el fusible proporciona un capa extra de protección. Si mas de 500mA son detectados en el puerto USB, el fusible automáticamente corta la conexión hasta que el cortocircuito o la sobretensión desaparece.

3.1.3. Motor, Ruedas y Soportes

Se han utilizado 2 motores EMG30 para la plataforma. El EMG30 se trata de un motor de corriente continua de 12V y 170 revoluciones que se caracteriza por incluir un encoder o codificador de cuadrante que manda un tren de impulsos cuando gira el eje del motor, permitiendo así que un circuito externo pueda saber la velocidad real a la que esta girando el eje y cuantas vueltas da. El encoder está formado por dos sensores de efecto hall que proporcionan un total de 360 pulsos por cada vuelta completa del rotor. El motor cuenta con condensadores internos de filtro que ayudan a minimizar el ruido y los parásitos generados por el motor al girar. En el Anexo A se encuentran las especificaciones completas del motor ofrecidas por el fabricante.

Figura 3.1.4: Motor EMG30

En cuanto a las ruedas se tratan de ruedas de goma con un diametro de 12 cm que incluyen un casquillo de aluminio para el eje del motor de 5mm.

Figura 3.1.5: Rueda de goma

Y para facilitar el montaje de los motores se han incluido el soporte que se muestra en la siguiente figura, que está fabricado en alimunio de 2mm de espesor y con acabado en azul. En el anexo A se muestran las medidas del soporte ofrecidas por el fabricante.

Figura 3.1.6: Soporte para el motor

3.1.4. Controlador de Motores

El circuito de control de potencia para los motores es un L298N que incluye dos puentes H completos de hasta 2A. Si bien los motores pueden requerir más corriente (hasta 2.5A) en picos, el disipador incluído en el módulo le permite funcionar sin calentamiento y se obtienen pares suficientes para esta aplicación.

Cada puente H se controla con tres líneas: EN1(2) (Enable), conectada a una salida PWM; IN1(3) e IN2(4) que seleccionan la polaridad (dirección) y el tipo de parada (libre o con freno). Se requieren por tanto seis salidas desde el Arduino hasta el módulo del L298N.

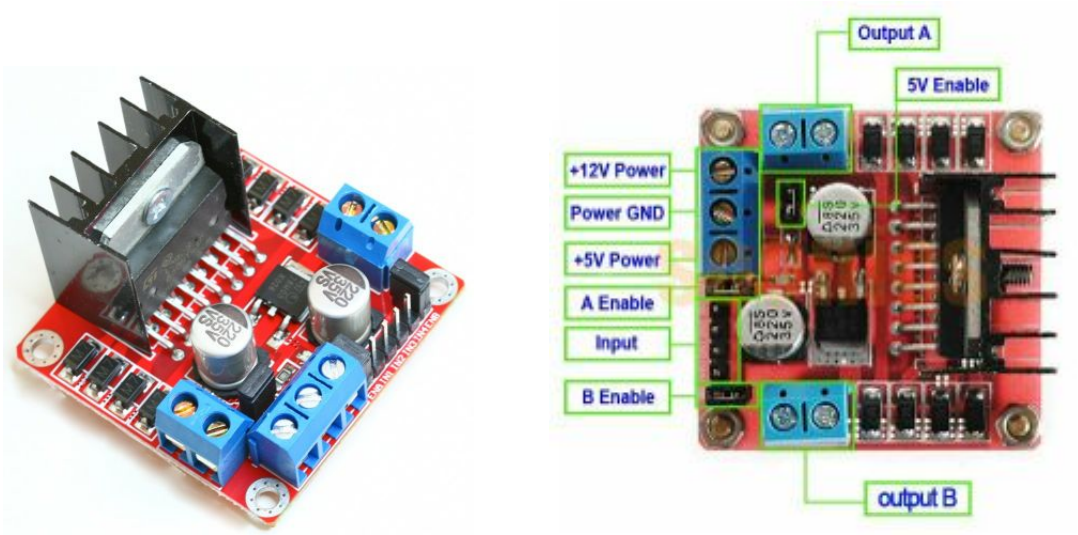


Figura 3.1.7: L298N

En la siguiente tabla se muestran las características ofrecidas por el fabricante:

Chip del controlador	chip de doble puente H L298N
Alimentación en terminales VMS	5 V - 35 V
Pico de corriente en terminales Io	2A / puente
Suministro en terminal Vss	4.5-5 .5 V
Funcionamiento del terminal	0 - 36 mA
Señal de control de voltaje de entrada	Alto 4.5-5.5V, Bajo 0V.
Consumo máximo de energía	25W
Temperatura de almacenamiento	-25 ℃ - 130 ℃
Tamaño del controlador	43x43x27mm
Peso	30g

Tabla 3.2: Características L298N

3.1.5. Sensor de Distancias

En este proyecto se ha utilizado el sensor de distancias ultrasónico HC-SR04. El sensor está formado por un emisor y un receptor de ultrasonidos que opera a una frecuencia de 40kHz. Es un sensor de 4 pines, dos de alimentación –VCC y GND- y dos para capturar la distancia –Trig y Echo-.

Figura 3.1.8: HC-SR04

El sensor se alimenta a 5Vdc, por lo que se puede alimentar directamente desde Arduino, y se puede llegar a detectar objetos hasta 5m con una resolución de 1cm. Para medir la distancia con el HC-SR04 hay que generar un pulso en el pin Trig de un ancho o tiempo de 10µs como mínimo. Al mismo tiempo hay que monitorizar la señal que llega al pin de Echo. La distancia calculada por el sensor se corresponde a la formula:

$$distancia = \frac{(Ancho\,de\,pulso \times Velocidad\,sonido)}{2}$$

Se ha incluido en el Anexo B de este proyecto la hoja de especificaciones del sensor ofrecida por el fabricante.

3.1.6. Conector Bluetooth

Como conector bluetooth se ha elegido el modelo HC-06. Se trata de un conector bluetooth bastante económico que trabaja mediante conexión serie con el Arduino. Dispone de un regulador de 3.3 v por lo que se puede conectar directamente a los 5 v del Arduino.

Figura 3.1.9: Módulo bluetooth

Como se observa en la figura previa, solo dispone de 4 pines que son los necesarios, por un lado VCC que irá conectado a los 5V del Arduino, por otro GND que irá al pin GND de Arduino, después TXD que irá conectado al pin RXD del Arduino y RXD que irá al pin TXD de Arduino. Se ha includo el manual de usuario de este conector en el Anexo C del presente proyecto.

3.1.7. Bateria

Se han usado baterías de tipo LIPO de 3 celdas (11.1V) de 2200 mAh por la corriente que proporcionan y su carga rápida. Proporcionan energía a la electrónica de potencia de los motores y a al Arduino que reduce la tensión a 5V mediante un regulador propio. La linea de tierra de las baterías es común a todo él sistema electrónico del robot.

Figura 3.1.10: Batería LIPO de 3 celdas

3.2. Distribución y Montaje

Uno de los objetivos del presente proyecto era la elección de la distribución de los diferentes componentes sobre la plataforma. Los componentes que tenían un posición establecida en el punto de partida del proyecto (debido al diseño de partida) erán los motores con sus respectivos soportes, la rueda loca pasiva y los soportes para sensores en la parte delantera. Los componentes que debían ser distribuidos son: la batería, el Arduino mega, el controlador de motores y el conector bluetooth.

Todos los componentes relacionados con la electronica de potencia (batería y controlador de motores) se situaron en la parte inferior. En la parte central inferior se sitúo el controlador de motores, para la sujeción se utilizaron tornillos (M2.5) con separadores de 5 mm de nylon, con el objetivo de que el controlador no se encontrara en contacto directo con la plataforma. En la siguiente figura se puede observa el componente montado sobre la ella:

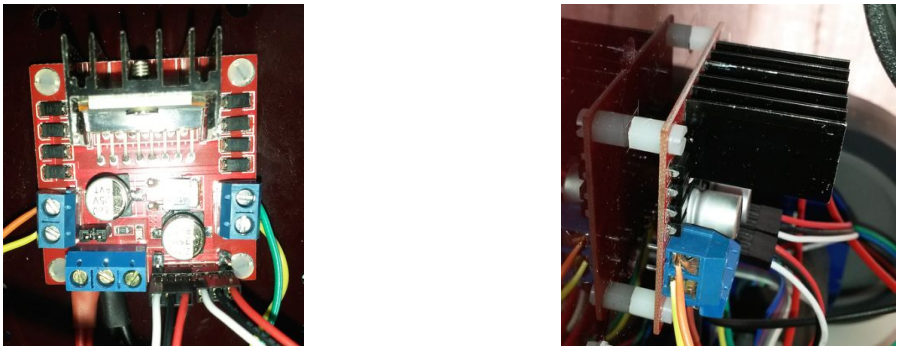


Figura 3.2.1: Localización del L298N

La batería se ha situado en la parte trasera de la plataforma, justo detrás de los motores.

Para su sujeción se le han colocado dos placas metálicas de 60 x 15 mm sobre la batería. Para ello se han utilizado abrazaderas de nylon de 3.56 mm de ancho. En la siguiente figura se puede observar el resultado:

Figura 3.2.2: Abrazaderas sobre la bateria

Y con el objetivo de poder quitar la batería de la plataforma de manera fácil, con solo 2 tornillos, se le ha añadido a la platforma dos escuadras metálicas de 30 x 30 mm. En la siguiente figura se observa, a la izquierda la escuadra utilizada y a la derecha la batería montada sobre la plataforma:

Figura 3.2.3: Montaje de la batería sobre la plataforma

Para la conexión de la batería se ha utilizado un conector XT60 hembra, un interruptor y un conector DC de 5.5 x 2.1 para la conexión con el arduino. El esquema de conexión de la batería se muestra en la siguiente figura:

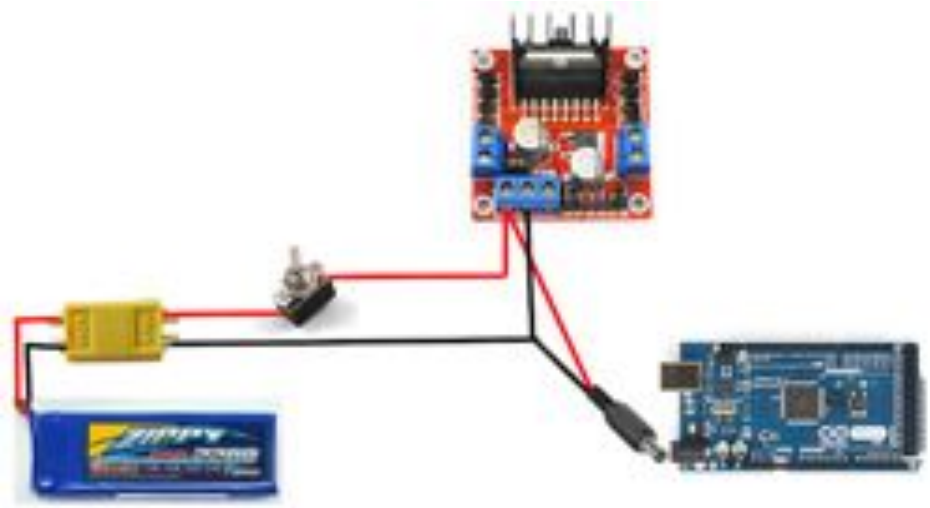


Figura 3.2.4: Esquema de conexión

De manera que la tierra de la batería es común a todo el sistema electrónico y el arduino reduce la tensión a 5V mediante el regulador propio.

En cuanto a las conexión con los motores, se ha utilizado un conector macho JST-PH de 6 vias para la conexión con el conector de los motores y en el otro extermo del cable, 4 vias van a la conexión con la shield para leer los encoders utilizando una cabecera tipo dupont, y las dos vías restantes van al controlador de motores L298N. El la siguiente figura se muestra un esquema de este conexionado:

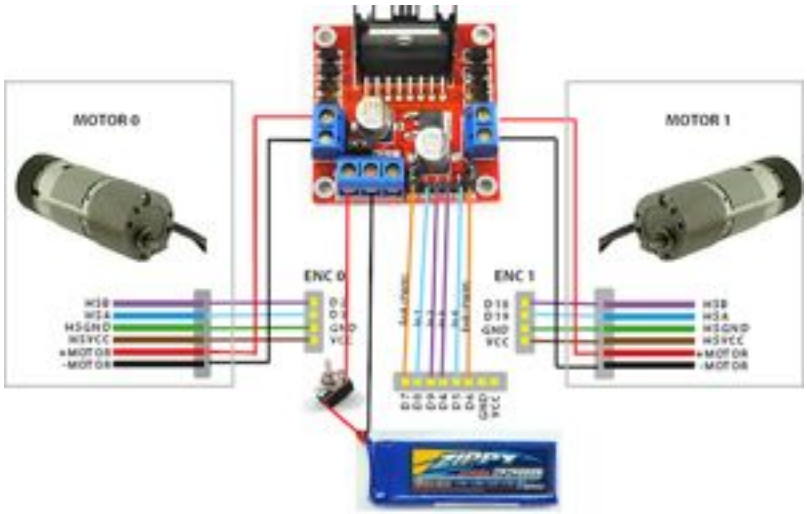


Figura 3.2.5: Esuqema de conexión de los motores

El Arduino Mega se ha situado en la parte superior, para ello se han utilizado tornillos (M2.5) y separadores de nylon, para que no este en contacto directo con la plataforma. En

la siguiente captura se muestra en detalle su posición con la shield de conexiones montada sobre él:

Figura 3.2.6: Situación Arduino Mega

Como punto de partida del proyecto además de la platafarma también se diseñaron previamente unos soportes compatibles para sensores como el sensor de distancias óptico sharp. Sin embargo el conector bluetooth no se puede montar sobre este tipo de soporte. Es por ello que se utilizó una placa de prototipos stripboard de 1.4 x 3.8 mm, para soldar en vertical sobre ella el conector bluetooth y también un conector de 4 pines de 0.1". A esta placa se le añadieron dos orificios para tornillos M3 para poder sujetarla a la parte trasera de la plataforma. En la siguiente figura se observa el resultado:

Figura 3.2.7: Soporte conector bluetooth

Para finalizar en la siguientes figuras se muestra el montaje completo de la plataforma, a la que además se le ha añadido el sensor de distancias ultrasónico. Este sensor esta sujeto al soporte de sensores con tornillos de nylon M02 y separadores M02 de 1 cm de longitud:

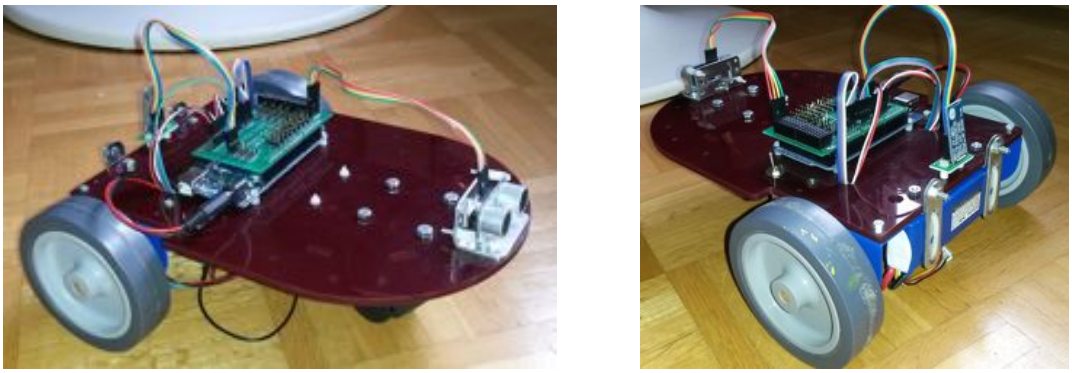


Figura 3.2.8: Soporte conector bluetooth

Capítulo 4

PCB Shield de Conexiones

Una placa de circuito impreso (PCB, "Printed Circuit Board") es una superficie constituida por caminos o pistas de material conductor (generalmente cobre) laminadas sobre un sustrato no conductor (cómo fibra de vidrio o baquelita). Se utiliza para conectar eléctricamente a través de las pistas y sostener mecánicamente mediante el sustrato un conjunto de componentes electrónicos.

Las placas de circuito impreso son el método más perfeccionado y que ofrece el acabado más fiable en la implementación de circuitos electrónicos, por el contrario su fabricación exige un proceso más laborioso. Las PCBs pueden ser de simple o doble cara, según alberguen pistan conductoras en un lado o ambos. También existen las PCB multicapa, que se fabrican mediante la superposición de láminas conductoras y aislantes.

En la realización del presente proyecto se ha diseñado e implementado una PCB. Esta PCB se trata de una «shield» que encaja sobre el Arduino Mega y su objetivo principal es poder facilitar las conexiones de los diferentes sensores y actuadores utilizados en la plataforma.

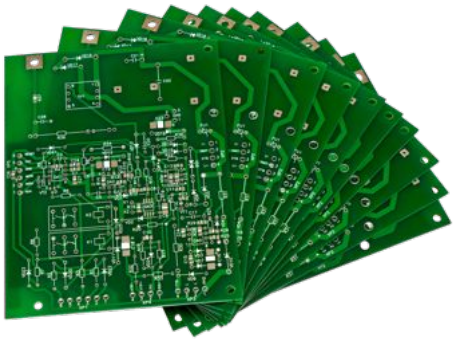


Figura 4.0.1: PCB

4.1. Requisitos

En este apartado se describen los requisitos específicos para el diseño de la PCB. El principal requisito a tener en cuenta es que se trata de una «shield» para Arduino Mega, por lo tanto se debe diseñar de manera que esta encaje perfectamente en los pines de conexión del Arduino. El segundo requisito es que esta «shield» de conexiones debe incluir una serie de puertos analógicos y digitales (incluyendo los pines VCC y GND) para facilitar la conexión con los sensores y actuadores compatibles con Arduino. En el siguiente listado se especifican los diferentes puertos que deben aparecer en la «shield»:

- 8 puertos analógicos dobles con conector tipo dupont (2 pines digitales, vcc y gnd).
- 8 puertos digitales dobles con conector tipo dupont (2 pines analógicos, vcc y gnd).
- 2 puertos de encoder (2 pines digitales conectados a líneas de interrupción externas, vcc y gnd).
- 1 puerto de mando de motores (pines ENA y ENB conectados a lineas PWM, 4 pines digitales, vcc y gnd).
- 3 puertos UARTs (TX, RX, vcc y gnd).
- 4 puertos I2C (SDA, SCL, vcc y gnd).

4.2. Diseño de la PCB

El software que se ha utilizado para el diseño de la placa de circuito impreso ha sido Eagle en concreto su versión Eagle 6.5.0. Eagle (siglas de "Easily Applicable Graphical Layout Editor") es una herramienta de fabricación de circuitos impresos que incorpora autoenrutador, desarrollada por Cadsoft. Famoso alrededor del mundo de los proyectos electrónicos, posee las ventajas de que muchas versiones de este programa tienen una licencia Freeware y gran cantidad de bibliotecas de componentes alrededor de la red.

Se ha diseñado una «shield» de tamaño 8.5 cm x 5.3 cm de doble capa. En este diseño se ha utilizado la librería de componentes de «SparkFun» [1] que proporciona componentes varios agrupados por categorías (resistencias, condensadores, esquemas de arduino....). En la siguiente imagen se puede observar el diseño final de esta shield:

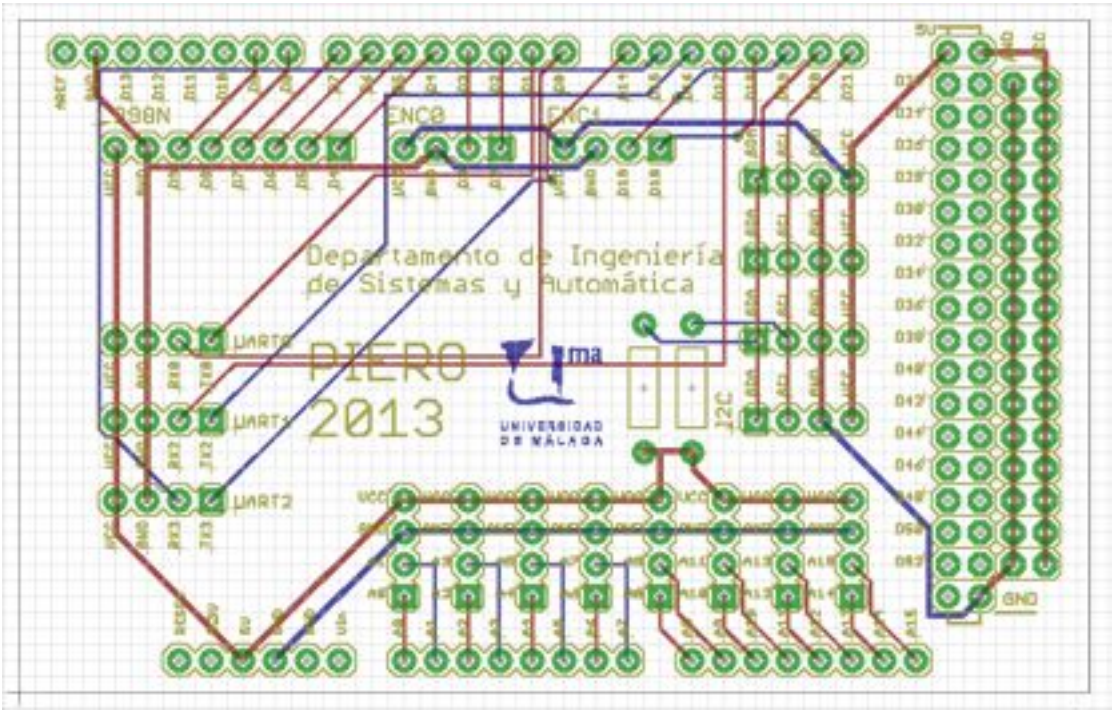


Figura 4.2.1: Diseño Final

En la parte inferior del diseño se han añadido los 8 puertos analógicos con dos pines analógicos cada uno, y donde se incluyen los pines vcc y gnd. Por lo tanto cada puerto analógico consta de 4 pines. En el diseño se ha elegido el componente de la librería SparkFun M04PTH que se trata de un conector de cuatro pines de 0.1".

En cuanto al I2C, en el Arduino Mega los pines I2C son el pin digital 21 para SCL y el pin digital 20 para SDA. Es por ello que se han distribuido los cuatro puertos I2C requeridos en este proyecto en una zona cercana a estos pines. Los conectores utilizados al igual que el caso de los puertos analógicos han sido el modelo M04PTH. El Arduino Mega incluye resistencias pull-ups de 20k incluidas en el chip Atmega, sin embargo se ha decidido incluir un soporte para resistencias conectadas a los pines SDA y SCL y a vcc. De esta manera pueden ser utilizadas resistencias pull-up con diferentes valores sin tener que utilizar las integradas en el chip Atmega. Para las resistencias se ha elegido el componente RESISTORPTH-1/4W.

Para el puerto de mando de motores se han elegido los pines digitales del 4 al 9, todos ellos son pines PWM. Se ha etiquetado con el nombre «L298N» y se ha situado en una posición cercana a estos pines. Además se han añadido dos pines para vcc y gnd obteniendo un puerto de 8 pines. El conector utilizado ha sido el modelo M081x08 que se trata de un conector de ocho pines de 0.1".

Los dos puertos con encoders necesitan pines digitales con interrupciones reales. El Arduino Mega ofrece los siguientes pines con interrupciones reales 2, 3, 18, 19, 20 y 21. Sin

embargo los pines 21 y 20 son utilizados por el puerto I2C, por lo tanto se han elegido los pines 2 y 3 para el puerto etiquetado como «ENC0», y los pines 18 y 19 para el puerto «ENC1». Estos puertos se han situado en una posición cercana a estos pines y se ha utilizado el mismo conector de cuatro pines de 0.1" utilizado previamente (M04PTH).

Los puertos UARTs utilizados han sido el 0, el 2 y el 3 etiquetados en el diseño como UART0, UART1 y UART2. No se ha utilizado el puerto 1 porque utiliza los pines 18 y 19 que son utilizados en el puerto para los encoders «ENC1» mencionado previamente. Se han utilizado los mismos conectores de cuatro pines de 0.1", al incorporar los pines para vcc y gnd.

Para los pines digitales la especificación requería al menos 8 puertos digitales con dos pines digitales cada uno. Sin embargo para poder optimizar el espacio se ha incluido una tira de pines dobles para vcc y gnd en el lateral de los pines digitales del 21 al 53. De esta manera el diseño final cuenta con 16 puertos digitales de cuatro pines, 2 digitales, vcc y gnd.

En la serigrafía de la capa superior se han incluido etiquetas con los nombres de todos los pines, los nombres de los pines, los conectores y en la parte central del diseño se ha incorporado el nombre del departamento, el nombre del proyecto y el logo de la Universidad de Málaga.

El enroutado de los componentes se ha realizado utilizando el algoritmo de enroutado automático ofrecido por el software Eagle. En la imagen se puede observar en rojo las pistas de la capa superior y en azul las pistas de la capa inferior. En el anexo I, se encuentra el esquemático del diseño.

4.2.1. Cálculo del Ancho de las Pistas

Siguiendo el estándar general para el diseño de circuitos impresos, ANSI- IPC 2221, el cálculo del ancho de pista de los mismos se ha realizado siguiendo las Ecuaciones 4.2.1 y 4.2.2, en donde:

- I: Se corresponde con corriente máxima que pasará por la pista, en amperios (A).
- $\triangle T$: Se corresponde con el incremento de temperatura máximo, en grados centígrados (^{o}C).
- L: Se corresponde con el grosor de la capa de cobre, en onzas/píe2 (oz/ft^2). Salta a la vista el influjo de las medidas anglosajonas. Una pista con un grosor de 1 onza por píe cuadrado se corresponde con una pista de 35 micras de grosor. Normalmente las empresas que construyen circuitos impresos los hacen utilizando medidas estándar para el grosor de pista de 1, 2 y 3 onzas por píe cuadrado, es decir 35, 70 y 105 micras.

- $K1$: Se corresponde con la constante de 0,015 para pistas internas y 0,0647 para pistas externas.
- $K2$: Se corresponde con la contante de 0,5453 para pistas internas y 0,4281 para pistas externas.
- $K3$: Se corresponde con la constante de 0,7349 para pistas internas y 0,6732 para pistas externas.
- El resultado se tiene en milésimas de pulgada (mil), que en el caso de que se quiera pasar a milímetros (mm) se tendrá que tener en cuenta que 1 pulgada equivale a 25,4 milímetros.

$$Ancho = \frac{Area}{L{\cdot}1,378} \tag{4.2.1}$$

$$Area = \left(\frac{I}{K1{\cdot}\triangle T^{K2}}\right)^{\frac{1}{K3}} \tag{4.2.2}$$

A continuación se realizan los cálculos de anchos de pista para incrementos de temperatura permitidos de hasta 20ºC, siendo todas exteriores y teniendo en cuenta que el grosor de las capas exteriores es de 35 micras, es decir, 1 onza/pie2.

Para las pistas de potencia, considerando una corriente máxima de 2A:

$$Area = \left(\frac{2}{0{,}0647{\cdot}20^{0,4281}}\right)^{\frac{1}{0,6732}} = 24{,}3303\, mil^2$$

$$Ancho = \frac{24{,}33}{1{\cdot}1{,}378} = 17{,}6563\, mil$$

En el diseño final se han sobredimensionado estableciendo un valor de 20 mil para las lineas de potencia.

Para el resto de líneas, sabiendo que la intensidad máxima para cada E/S del Arduino es de 40mA:

$$Area = \left(\frac{0{,}04}{0{,}0647{\cdot}20^{0,4281}}\right)^{\frac{1}{0,6732}} = 0{,}0728\, mil^2$$

$$Ancho = \frac{0{,}0728}{1{\cdot}1{,}378} = 0{,}0529\, mil$$

En el diseño final se han sobredimensionado estableciendo un valor de 10 mil.

4.3. Fabricación de la Shield

Una vez terminado el diseño se mandó a fabricar un lote de 10 unidades. Para ello se seleccionó el fabricante «SeeedStudio». Las características de estas PCBs se describen en la siguiente tabla:

Número de PCBs	10
Número de Capas	2
Grosor de la PCB	1.6 mm
Color de la PCB	Verde
Superficie de Acabado	ENIG

Tabla 4.1: Características de la PCB

En la siguiente tabla se puede observar las especificaciones de manufactura ofrecidas por el fabricante:

Item	Specs	
	Unit: mm	Unit: mil
Available Board Thickness	0.6(4 layer except), 0.8, 1.0, 1.2, 1.6, 2.0	23.6(4 layer except), 31.5, 39.4, 47.2, 63.0, 78.7
Thickness Tolerance	(t >= 1.0) ± 10%	(t >= 39.4) ± 10%
Thickness Tolerance	(t < 1.0) ± 0.1%	(t < 39.4) ± 0.1%
Insulation Layer Thickness	0.075 - 5.0	2.95 - 196.85
Minimum trace width	0.1524	6
Minimum inner trace width(for 4 layer)	0.203	8
Minimum trace/vias/pads space	0.1524	6
Minimum inner trace/vias/pads space(for 4 layer)	0.203	8
Minimum silkscreen width	0.1524	6
Minimum silkscreen text size	0.8128	32
Out Layer Copper Thickness	> 0.03	> 1.18
Inner Layer Copper Thickness	0.01 - 0.018	0.39 - 0.71
Drilling Hole(Mechanical)	0.3 - 6.35	11.81 - 250.00
Finish Hole(Mechanical)	0.8 - 6.35	31.50 - 250.00
Diameter Tolerance(Mechanical)	± 0.2	± 7.87
SMT min Solder Mask Width	0.1	3.94
Min Solder Mask Clearance	0.13	5.12
Aspect Ratio	8:1	
Solder Mask Type	Photosensitive ink	

Tabla 4.2: Especificaciones de Manufactura

Para comprobar que el diseño cumple con estas especificaciones, el fabricante ofrece un fichero .dru que podemos utilizar con el software Eagle para comprobar si todo esta correcto o si hay que modificar alguna propiedad en la preferencias del diseño.

Para poder enviar el diseño hay que enviar al fabricante una serie de ficheros gerger que se pueden generar con el software Eagle y el procesador CAM que el fabricante ofrece. A continuación se listan los ficheros necesarios para fabricar una PCB de dos capas:

- Top Layer: pcbname.GTL
- Bottom Layer: pcbname.GBL
- Solder Mask Top: pcbname.GTS
- Solder Mask Bottom: pcbname.GBS
- Silk Top: pcbname.GTO
- Silk Bottom: pcbname.GBO
- Drill Drawing: pcbname.TXT
- Board Outline:pcbname.GML/GKO

Una vez generados estos ficheros con el diseño fueron enviados a fabricar. Un mes más tarde se recibieron las PCBs resultantes, en la siguiente imagen se puede observar el resultado:

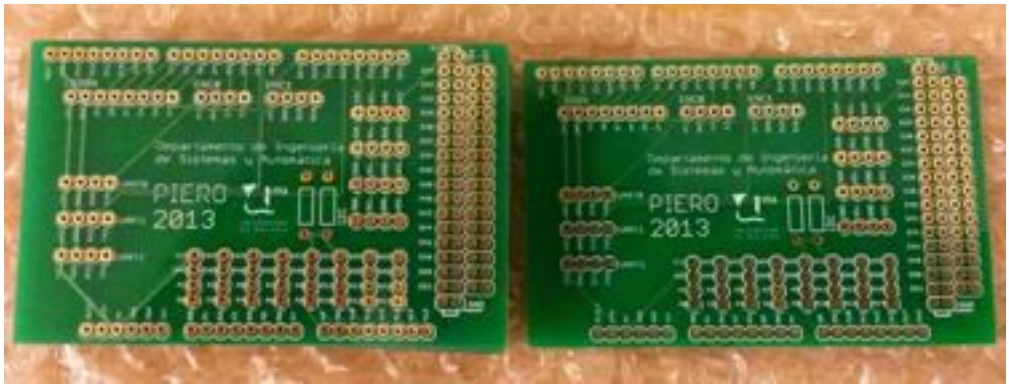


Figura 4.3.1: PCB Final

Para el conexionado se han utilizado 3 tipos de conectores. Para los puertos digitales se han utilizado conectores hembras con pines largos y para las tiras de vcc y gnd conectores hembra dobles (ambos con 0.1" de espaciado entre pines). Para el resto de puertos se han utilizado tiras de pines con 0.1" de espaciado. En las siguiente figura observamos estos tres tipos de conectores:

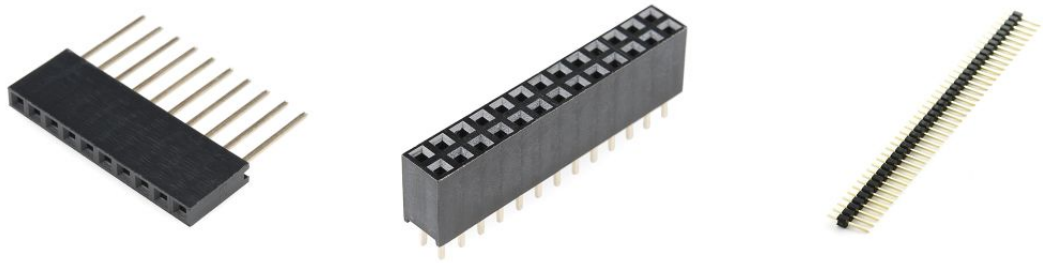


Figura 4.3.2: Conectores Utilizados

Y en la siguiente figura se puede observar el resultado final, la shield con los conectores soldados montada sobre el Arduino y sobre la plataforma PIERO.

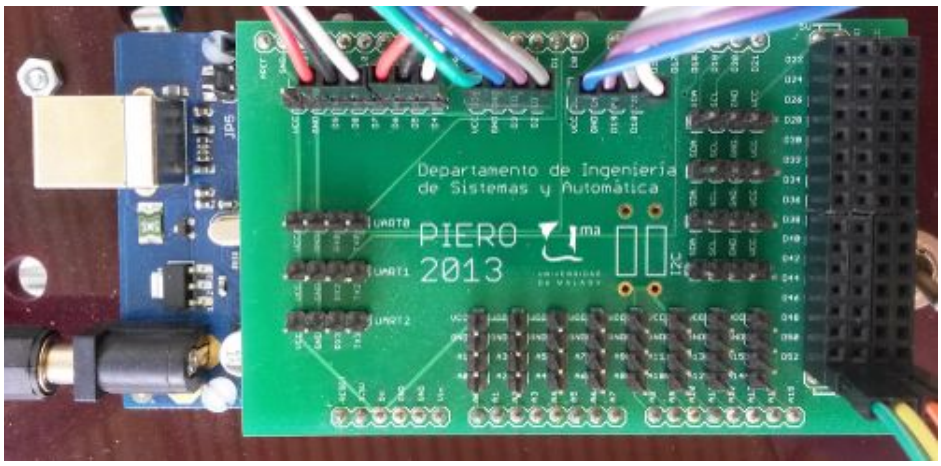


Figura 4.3.3: Shield sobre Arduino

Capítulo 5

Drivers de Arduino para Simulink

En este capítulo se va a describir el proceso de creación de drivers Arduino para Simulink, además se describirán las características de los diferentes drivers desarrollados en el presente proyecto, lo cuales serán utilizados de manera transparente en las prácticas descritas en el capítulo 6.

5.1. ¿Qué es un Driver?

De forma general en Simulink se considera un driver a un bloque que toma una señal como entrada y la utiliza para calcular algún tipo de actuación/operación. Dónde la entrada, la actuación o ambas se realizan sobre algún hardware externo. Por ejemplo la lectura de un sensor de distancia, la escritura en una pantalla LED, la lectura de un encoder....

Figura 5.1.1: Driver

5.2. Instalación del Soporte Arduino en Simulink

Para instalar el soporte Arduino en Simulink hay que seguir los siguientes pasos:

- En el terminal de comandos de Matlab hay que teclear el siguiente comando (que inicializa el instalador):

```
>> supportPackageInstaller
```

- A continuación se puede selecionar el origen del paquete. Lo más recomendable es obtener el paquete desde los servidores de Mathwork, para eso hay que asegurarse que se dispone de una conexión a internet y seleccionar instalar desde internet:

Figura 5.2.1: Fuente del Paquete

- Posteriormente se debe selecionar Arduino en la lista de componentes disponibles y pulsar siguiente:

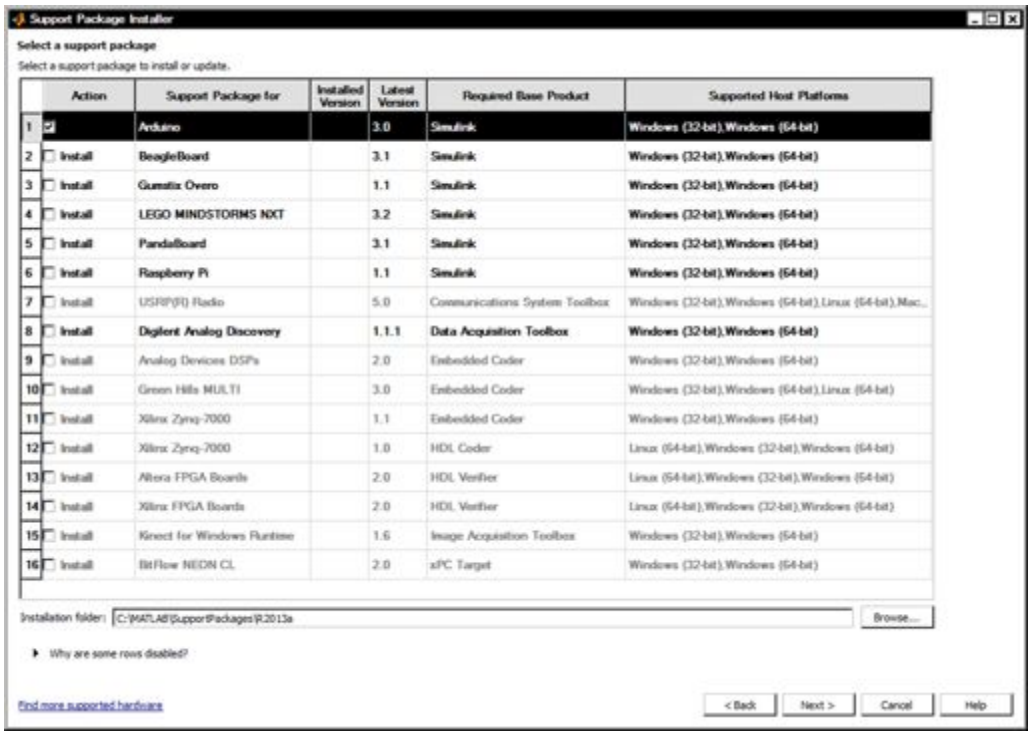

Support Package Installer
Select a support package
Select a support package to install or update.
Action
Support Package for
Installed Version
Latest Version
Required Base Product
Supported Host Platforms
1 Arduino 3.0 Simulink Windows (32-bit),Windows (64-bit)
2 Install BeagleBoard 3.1 Simulink Windows (32-bit),Windows (64-bit)
3 Install Gumstix Overo 1.1 Simulink Windows (32-bit),Windows (64-bit)
4 Install LEGO MINDSTORMS NXT 3.2 Simulink Windows (32-bit),Windows (64-bit)
5 Install PandaBoard 3.1 Simulink Windows (32-bit),Windows (64-bit)
6 Install Raspberry Pi 1.1 Simulink Windows (32-bit),Windows (64-bit)
7 Install USRP(R) Radio 5.0 Communications System Toolbox Windows (32-bit),Windows (64-bit),Linux (64-bit),Mac...
8 Install Digilent Analog Discovery 1.1.1 Data Acquisition Toolbox Windows (32-bit),Windows (64-bit)
9 Install Analog Devices DSPs 2.0 Embedded Coder Windows (32-bit),Windows (64-bit)
10 Install Green Hills MULTI 3.0 Embedded Coder Windows (32-bit),Windows (64-bit),Linux (64-bit)
11 Install Xilinx Zynq-7000 1.1 Embedded Coder Windows (32-bit),Windows (64-bit)
12 Install Xilinx Zynq-7000 1.0 HDL Coder Linux (64-bit),Windows (32-bit),Windows (64-bit)
13 Install Altera FPGA Boards 2.0 HDL Verifier Linux (64-bit),Windows (32-bit),Windows (64-bit)
14 Install Xilinx FPGA Boards 2.0 HDL Verifier Linux (64-bit),Windows (32-bit),Windows (64-bit)
15 Install Kinect for Windows Runtime 1.6 Image Acquisition Toolbox Windows (32-bit),Windows (64-bit)
16 Install BitFlow NEON CL 2.0 xPC Target Windows (32-bit),Windows (64-bit)
Installation folder: C:\MATLAB\SupportPackages\R2013a
Browse...
Why are some rows disabled?
Find more supported hardware
< Back
Next >
Cancel
Help

Figura 5.2.2: Listado de Componentes

- Y por último se debe introducir los datos de una cuenta Mathwork válida y esperar a que termine el instalador:

Figura 5.2.3: Login Cuenta de Mathwork

5.3. Arduino en Simulink

El paquete «Arduino Support from Simulink» [3] nos permite utilizar Simulink para crear algoritmos que serán compilados y ejecutados en la plataforma Arduino, esto se realiza de manera transparente de cara al usuario. Además este paquete nos ofrece una librería con una serie de drivers para realizar operaciones básicas en el Arduino. En la siguiente tabla se observan los diferentes bloques que ofrece esta librería:

Bloque	**Descripción**
Arduino Analog Input	Mide la tensión de una entrada analógica.
Arduino Continuous Servo Write	Ajusta la velocidad de rotación de un servo motor.
Arduino Digital Input	Obtiene el valor lógico de una entrada digital.
Arduino Digital Output	Establece el valor lógico de una salida digital.
Arduino PWM	Genera una señal PWM en una salida analógica.
Arduino Serial Receive	Recibe un byte a través del puerto serie.
Arduino Serial Transmit	Envía un conjunto de bytes a través del puerto serie.
Arduino Standard Servo Read	Obtiene la posición de rotación de un servo motor en grados.
Arduino Standard Servo Write	Establece la posición de un servo motor.
Arduino TCP/IP Receive	Recibe un mensaje TCP/IP en la red cableada.
Arduino TCP/IP Send	Envía un mensaje TCP/IP desde el servidor.
Arduino UDP Receive	Recibe un mensaje UDP.
Arduino UDP Send	Envía un mensaje UDP a una interfaz remota.
Arduino WiFi TCP/IP Receive	Recibe un mensaje WiFi TCP/IP en la red inalámbrica.
Arduino WiFi TCP/IP Send	Envía un mensaje WiFi TCP/IP desde el servidor.
Arduino WiFi UDP Receive	Recibe un mensaje WiFi UDP.
Arduino WiFi UDP Send	Envía un mensaje WiFi UDP a una interfaz remota.

Tabla 5.1: Bloques Librería Arduino de Simulink

Para poder consultar en profundidad los detalles de cada uno de los bloques listados en la tabla anterior, se puede consultar el apartado de la web de Mathwork dedicado al mismo [2].

Inicialmente parece que este paquete ofrece una gran variedad de drivers para diversas utilidades, pero rápidamente puede ser detectada que existen limitaciones que pondrían en una severa desventaja al programador Simulink con respecto al programador nativo de Arduino. La primera gran limitación es la imposibilidad de utilizar interrupciones, algo completamente necesario, para por ejemplo poder leer el encoder de un motor. Otra gran limitación es la imposibilidad de utilizar funciones desarrolladas de manera muy eficiente en algunas librerías externas a Arduino.

Pero gracias a que podemos desarrollar driver específicos, éstas dos grandes limitaciones desaparecen. Esto se debe a que haciendo uso de los bloques «S-Function» podemos crear bloques Simulink que contiene código Matlab/C/C++ e incluso utilizar librerías C++ externas desarrolladas para Arduino. Y estos bloques serán compilados y ejecutados en el Arduino de manera automática por Simulink.

5.4. Guía de Creación de Drivers

En esta sección se resume la guía para crear drivers proporcionada por Mathwork [9], de manera que se centrará solo en los apartados necesarios en el presente proyecto. Se utilizará el bloque de Simulink «S-Function Builder» [13], que se encuentra en la librería de Simulink bajo la opción «User-Defined Functions» y permite generar S-Functions (concretamente, un fichero C «wrapper» que continente solo información de cabeceras y dos funciones) sin tener que crear un fichero C que utilice de manera explícita su API. El fichero «wrapper» se utilizará posteriormente para crear los ejecutables para la simulación o para ser ejecutados en una plataforma externa como Arduino.

Para comenzar, se crea un nuevo modelo Simulink y se añade un bloque «S-Function Builder». El siguiente paso es seleccionar la plataforma de destino para el modelo. Esto se lleva a cabo seleccionando «Tools->Run on Target Hardware->Prepare to Run». Posteriormente se define el nombre de la S-Function. En los ejemplos mostrados a continuación se utiliza el nombre «sfcn_exout_slsp» en la ventana de opciones, que se despliega al hacer doble-click sobre el bloque. A continuación se pasa a la segunda pestaña «Data Properties»:

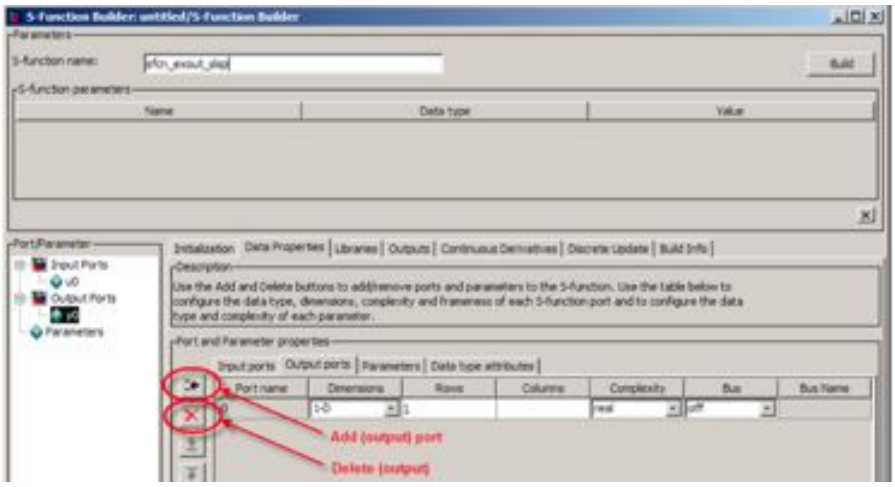


Figura 5.4.1: Data Properties

Esta pestaña permite definir el número y la dimensión de los puertos de entradas y salidas, así como los parámetros que se le pasan al bloque S-Function. En la pestaña «Data type attributes» se encuentran las opciones para gestionar los tipos de datos de las entradas, salidas y parámetros. Cuando se añaden parámetros, el valor de los mismos puede ser modificado en el listado que aparece debajo del nombre de la función, etiquetado como «S-function parameters». Hay que tener en cuenta que cada vez que se modifiquen los valores de los parámetros hay que recompilar la S-Function para que este cambio tenga efecto.

En la última pestaña etiquetada como «Build Info», se debe marcar la opción «Generate wrapper TLC». Esto generará un fichero TLC que será utilizado para compilar el ejecutable que se desplegará en la plataforma objetivo. En la siguiente figura se observa esta opción:

Figura 5.4.2: Generación del Fichero TLC

En la pestaña «Initialization» es establece el tiempo de sampleo del bloque y el número de estados continuos y discretos. Normalmente los bloques se ejecutaran en tiempo discreto y sin estados continuos. En este caso, como se observa en la siguiente figura, se ha seleccionado un tiempo de sampleo de 50ms.

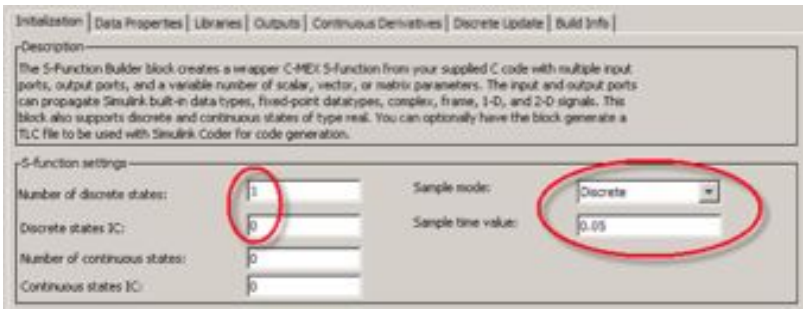

Initialization
Data Properties
Libraries
Outputs
Continuous Derivatives
Discrete Update
Build Info
Description
The S-Function Builder block creates a wrapper C-MEX S-function from your supplied C code with multiple input ports, output ports, and a variable number of scalar, vector, or matrix parameters. The input and output ports can propagate Simulink built-in data types, fixed-point datatypes, complex, frame, 1-D, and 2-D signals. The block also supports discrete and continuous states of type real. You can optionally have the block generate a TLC file to be used with Simulink Coder for code generation.
S-function settings
Number of discrete states:
1
Discrete states IC:
0
Number of continuous states:
0
Continuous states IC:
0
Sample mode:
Discrete
Sample time value:
0.05

Figura 5.4.3: Panel de Inicialización

La implementación de un driver requiere que se establezca al menos un estado de tiempo discreto. Se podrían añadir más estados en el caso de necesitarlos, pero el primer elemento del vector de estados discretos (que es xD[0]) es inicializado a 0, hecho que permite utilizarlo para ejecutar código de inicialización. La pestaña «Discrete Updata» define en general las leyes de evolución del vector de estados; sin embargo, como se muestra en la siguiente figura, aquí se utiliza para ejecutar código de inicialización. Este se escribe directamente en campo de edición de texto:

Initialization
Data Properties
Libraries
Outputs
Continuous Derivatives
Discrete Update
Build Info
Code description
This section is optional and use to update the discrete states. It is called only if the S-function has one or more discrete states. The states of the S-function are of type double and must be referenced as xD[0], xD[1], etc. respectively. Input ports, output ports and parameters should be referenced using the symbols specified in the Data Properties. These references appear directly in the generated S-function.
if (xD[0]!=1) {
/* don't do anything for MEX-file generation */
#ifndef MATLAB_MEX_FILE
pinMode(pin[0],OUTPUT);
#endif
/* initialization done */
xD[0]=1;
}

Figura 5.4.4: Código de Inicialización

Tal como se puede comprobar en el trozo de código mostrado, dado que el valor inicial del vector de estados es 0, el contenido del «if» solo se ejecutará una vez, por eso en la última línea se modifica dicho valor y se establece a 1. La directiva *#ifndef MATLAB_MEX_FILE* se utiliza para evitar que este código se compile para su simulación en Windows, en cambio cuando se compile para Arduino no estará definida la etiqueta *MATLAB_MEX_FILE* por lo tanto si que se incluirá el código contenido (*pinMode([0),OUTPUT*).

Este código llama a la función arduino «pinMode» que establecerá el modo del pin especificado por el parámetro pin[0] a «OUTPUT». Para más información se puede consultar la referencia de Arduino [11]. De esta forma este mismo código puede ser utilizado para la definición de cualquier driver, donde en el interior de las directivas *#ifndef* y *#endif* se introducirá el código de inicialización del mismo.

En la pestaña «Outputs update» se definen las acciones que el bloque lleva a cabo en las salidas cuando éste es ejecutado. Como podemos observar en el código que se muestra en la siguiente figura:

Initialization | Data Properties | Libraries | Outputs | Continuous Derivatives | Discrete Update | Build Info
Code description
Enter your C-code or call your algorithm. If available, discrete and continuous states should be referenced as, xD[0]...xD[n], xC[0]...xC[n] respectively. Input ports, output ports and parameters should be referenced using the symbols specified in the Data Properties. These references appear directly in the generated S-function.
/* wait until after initialization is done */
if (xD[0]==1) {
/* don't do anything for mex file generation */
# ifndef MATLAB_MEX_FILE
digitalWrite(pin[0],in[0]);
# endif
}

Figura 5.4.5: Outputs update

ahora la condición es la contraría a la que se definió en el panel «Discrete Update», por lo tanto este código no se ejecutará hasta que no se haya inicializado el sistema.

La última pestaña que se tendrá en cuenta en este manual es «Libraries». Esta pestaña permite al usuario especificar librerías externas, incluir ficheros, variables globales y las funciones externas que sean necesarias en el código escrito en los otros paneles. En la siguiente figura se puede comprobar cómo incluir la librería «Arduino.h» que contiene entre otras cosas, las declaraciones de las funciones «pinMode» y «digitalWrite» usadas en los paneles «Discrete» y «Output».

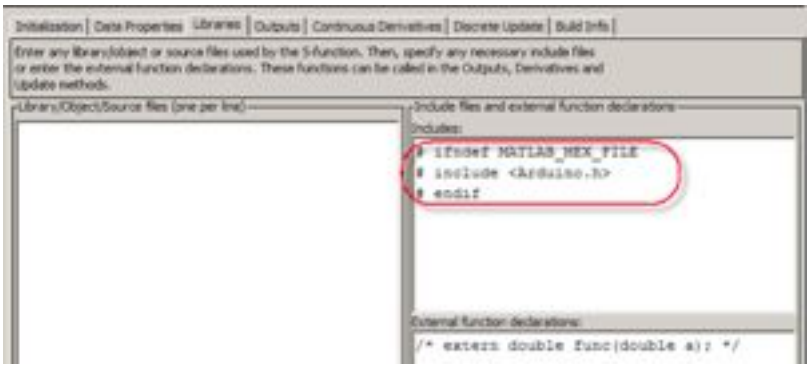

Initialization | Data Properties | Libraries | Outputs | Continuous Derivatives | Discrete Update | Build Info
Enter any library/object or source files used by the S-function. Then, specify any necessary include files or enter the external function declarations. These functions can be called in the Outputs, Derivatives and Update methods.
Library/Object/Source files (one per line)
Include files and external function declarations
Includes:
ifndef MATLAB_MEX_FILE
include <Arduino.h>
endif
External function declarations:
/* extern double func(double a); */

Figura 5.4.6: Librerías

Una vez incluido todo el código y establecidas todas las opciones se puede pulsar el botón «Build» y si todo va correcto, seis nuevos ficheros serán generados:

- Fichero «wrapper» (sfcn_exout_slsp_wrapper.c)
- Fichero S-Function para simulación (sfcn_exout_slsp.c)
- Fichero Mex (sfcn_exout_slsp.mexw64)
- Dos ficheros de configuración (rtwmakecfg.m y SFB_sfcn_exout_slsp_SFB.mat)
- Fichero TLC (sfcn_exout_slsp.tlc)

Finalmente y una vez compilada la S-Function, para poder enviar el código al Arduino se debe modificar el fichero «wrapper», modificando su extensión «.c» por «.cpp», y modificando en su interior la definición las funciones «void ... _Update_wrapper (const ...)» y «void ... _Outputs_wrapper (const ...)» añadiendo delante de la directiva «void» el siguiente texto: «extern "C"».

5.5. Bloques Desarrollados en el Proyecto

En esta sección se van a describir los diversos bloques drivers simulink desarrollados y utilizados en el presente proyecto.

5.5.1. Bloque DualQuadEncoder

El bloque DualEncoder se encarga de leer la posición y la velocidad de los dos motores incluidos en la plataforma, utilizando para ello los encoders incorporados en los mismos. Antes de describir en detalle las características del bloque se va a describir de manera breve el funcionamiento de los encoders.

Un encoder es un transductor rotativo, que mediante una señal eléctrica (normalmente un pulso o una señal senoidal) nos indica el ángulo girado. Si este sensor rotatorio lo conectáramos mecánicamente con una rueda o un husillo, también nos permitiría medir distancias lineales. Una clasificación de los encoder según el tipo de información sobre la posición que generan sería:

- Encoder incremental: La señal de salida se transmite por un hilo en el que se transmite un pulso por cada ángulo girado, de tal forma que si tenemos un encoder de 360 ppr, como es el caso de los encoders utilizados, tendremos un pulso por cada grado. El inconveniente es que no disponemos de una referencia absoluta de la posición en la que se encuentra el eje.

- Encoder absoluto: La posición se da en valor absoluto mediante un bus paralelo. Es decir, que si tenemos un encoder de 256 posiciones, tendremos un bus de 8 líneas que nos indicaran en binario cual es su posición (normalmente estos transductores codifican la posición en código gray para evitar errores). El inconveniente de estos encoders es la cantidad de líneas que necesitamos leer y conectar y que debido a la complejidad del disco óptico que codifica las posiciones la resolución no suele ser muy elevada.

Los encoders incrementales son los más utilizados y extendidos, además son los que monta los motores utilizados en este proyecto. A continuación se va a describir el algoritmo genérico a seguir para poder leer estos encoders. En la siguiente figura se puede observar las señales obtenidas a lo largo del tiempo por los canales A y B procedentes de los encoders:

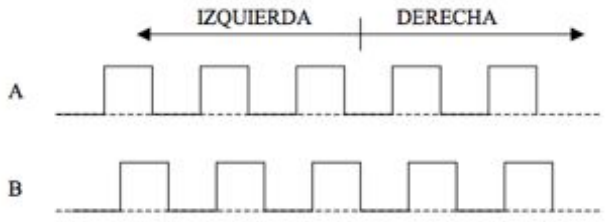


Figura 5.5.1: Señales encoders

Se trata de dos señales cuadradas que se encuentran desfasadas 90^{o}. En cada revolución completa del encoder se pueden contar 360 pulsos, con un flanco de subida (transición de 0 a 1) y un flanco de bajada (transición de 1 a 0) cada uno. Para determinar el ángulo girado por el encoder se cuenta el número de flancos de subida (o de bajada) generados en un canal (por ejemplo el A) y la dirección vendrá determinada, en cada pulso, por el estado del otro canal, de tal forma que si esta a 0 se determina que va en una dirección y si es 1 ira en la dirección contraria.

En la siguiente imagen se observa el bloque simulink creado:

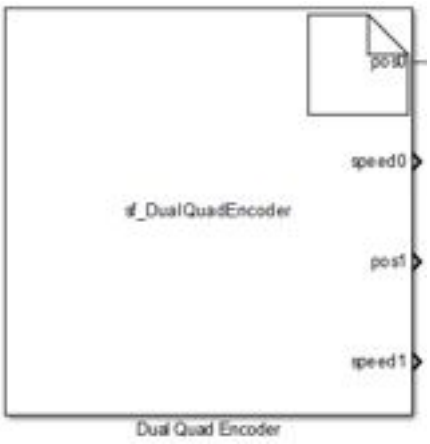


Figura 5.5.2: Bloque Simulink Encoder

Como se puede observar dispone de 4 salidas, dos para la posición y otras dos para la velocidad de cada motor. Este bloque dispone de 4 parámetros para poder indicar las entradas a las que están conectadas las salidas de los encoders de cada motor (las señales A y B de la figura 5.5.1). Este bloque tiene como requisito que estas entradas sean entradas paras las que el Arduino tenga asociada interrupciones. En el caso del Arduino Mega estas entradas son las siguientes: 2, 3, 18, 19, 20 y 21. En la siguiente figura se observan estos cuatro parámetros:

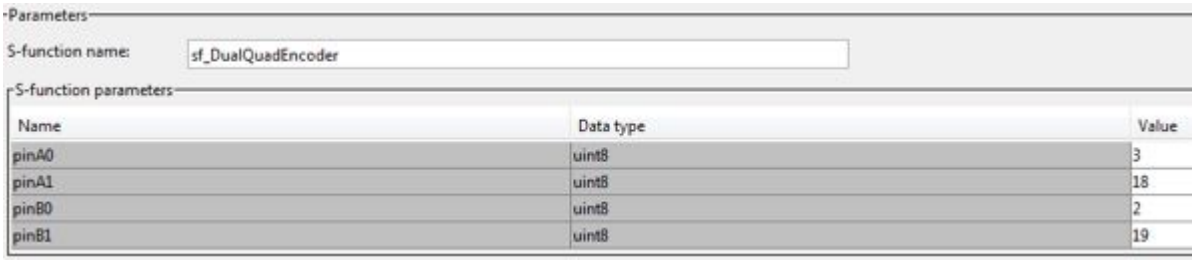


Figura 5.5.3: Parámetros Bloque Encoder

Este bloque lleva a cabo dos tareas, por una parte se encarga de llevar la cuenta de la posición de cada motor y por otra parte la velocidad. Para comenzar se va a describir cómo se consigue contabilizar la posición de los motores utilizando el algoritmo previamente descrito. El apartado de librerías contiene la definición de la estructura Encoder:

```
typedef struct {
        int pinA;
        int pinB;
        long posAnt;
        long pos;
        int speed;} Encoder;
```

Esta estructura servirá para almacenar información acerca del estado de cada encoder, donde «pinA» y «pinB» son las entradas, «posAnt» y «pos» son la posicion actual y la posición anterior y la propiedad «speed» almacena la velocidad del motor. También en este apartado se inicializa un array de tamaño dos que contendrá la información de los dos enconders:

```
volatile Encoder Enc[2] = {
        {0,0,0,0,0},
        {0,0,0,0,0}
};
```

Además en esta zona también se incluirán las definiciones de las funciones asociadas a cada interrupción. Estas funciones se han obtenido del código encontrado en el Matlab File Exchange [10]. Y se encargan de incrementar o decrementar la posición de los encoders en función del valor de las señales A y B.

En la zona de inicialización («Discrete Updates») se introduce el siguiente código, para establecer los valores de los pines A y B:

```
/* Establece pin A y B del motor 0 */
Enc[0].pinA = pinA0[0];
Enc[0].pinB = pinB0[0];

/* Establece pin A y B del motor 1 */
Enc[1].pinA = pinA1[0];
Enc[1].pinB = pinB1[0];
```

también se debe configurar estas señales como entradas y habilitar las resistencias pull ups, que se lleva a cabo con el siguiente código:

```
/* Configuramos los pines como entradas */
pinMode(Enc[0].pinA, INPUT);
pinMode(Enc[0].pinB, INPUT);
pinMode(Enc[1].pinA, INPUT);
pinMode(Enc[1].pinB, INPUT);

/* Habilitamos las resistencias pull ups. */
digitalWrite(Enc[0].pinA, HIGH);
digitalWrite(Enc[0].pinB, HIGH);
digitalWrite(Enc[1].pinA, HIGH);
digitalWrite(Enc[1].pinB, HIGH);
```

y se asocia las funciones de interrupción a cada pin:

```
attachInterrupt(getIntNum(Enc[0].pinA), irsPinAEn0, CHANGE);
attachInterrupt(getIntNum(Enc[0].pinB), isrPinBEn0, CHANGE);
attachInterrupt(getIntNum(Enc[1].pinA), irsPinAEn1, CHANGE);
attachInterrupt(getIntNum(Enc[1].pinB), isrPinBEn1, CHANGE);
```

La función «getIntNum» se encarga de obtener el número de la interrupción a partir del número del pin, esta se encuentra definida en la zona donde se incluyen las librerías. Por último, tan solo falta comentar el código que permite establecer el valor de las salidas. Este se define en la pestaña «Output» y lo único que hacer es leer la posición de las estructuras Encoders previamente definidas y volcarlo sobre las salidas.

```
pos0[0]=Enc[0].pos;
pos1[0]=Enc[1].pos;
```

Para medir la velocidad se ha utilizado la técnica basada en frecuencia [5], esta técnica consiste en obtener la posición de los motores cada cierto periodo de tiempo y calcular la velocidad utilizando la siguiente formula:

$$w\,(rpm) = \frac{\Delta pos \times 60}{T_m(s) \times E}$$

donde T_m representa el periodo de muestreo en segundos y E el número de pulsos por rotación que ofrece el encoder (360 en este caso). Y para seleccionar el periodo de muestreo mínimo se utiliza esta ecuación:

$$T_{min}\,(ms) > \frac{60 \times 1000}{w\,(rpm) \times E} = 166,66ms$$

El periodo utilizado en este proyecto ha sido 180 ms, para ello se ha configurado el timer5 del Arduino con este propósito. Cada 180 ms se ejecuta una función cuyo contenido se puede observar a continuación y se encuentra definida en la zona de definición de librerías del bloque s-function:

```
ISR(TIMER5_COMPA_vect){

        Enc[0].speed = (Enc[0].pos - Enc[0].posAnt)*60 / 0.18*360;
        Enc[0].posAnt = Enc[0].pos;

        Enc[1].speed = (Enc[1].pos - Enc[1].posAnt)*60 / 0.18*360;
        Enc[1].posAnt = Enc[1].pos;
}
```

lo que hace obtener la velocidad actual y almacenar la posición actual en la variable posAnt. Para realizar la cuenta de 180 ms se ha utilizado el timer 5 del ATmega2560 contenido en el Arduino Mega. Se trata de un temporizador de 16 bits que el arduino utiliza para generar las señales PWM en los pines 44, 45 y 46. Por lo tanto al utilizar este timer con otro propósito como es el caso del presente proyecto no se podrá generar PWMs a través de estas salidas.

Para poder utilizar este temporizador se necesita configurar una serie de regitros que los chips de AVR utilizan para almacenar las configuraciones de los temporizadores. Cada temporizador tiene una serie de registros para llevar a cabo diversas funciones. Dos de estos registros, que almacenan valores de configuración, son los llamados TCCRxA y TCCRxB. Donde x es el número del temporizador (en este caso sería TCCR5A y TCCR5B) y TCCR son las iniciales de «Timer/Counter Control Register». Cada registro tiene 8 bits y la utilidad de cada bit puede comprobarse en la siguiente figura:

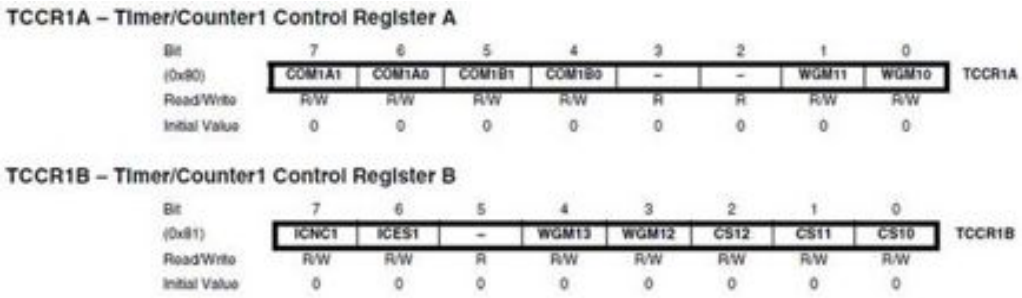

TCCR1A – Timer/Counter1 Control Register A
Bit 7 6 5 4 3 2 1 0
(0x80) COM1A1 COM1A0 COM1B1 COM1B0 – – WGM11 WGM10 TCCR1A
Read/Write R/W R/W R/W R/W R R R/W R/W
Initial Value 0 0 0 0 0 0 0 0
TCCR1B – Timer/Counter1 Control Register B
Bit 7 6 5 4 3 2 1 0
(0x81) ICNC1 ICES1 – WGM13 WGM12 CS12 CS11 CS10 TCCR1B
Read/Write R/W R/W R R/W R/W R/W R/W R/W
Initial Value 0 0 0 0 0 0 0 0

Figura 5.5.4: Registros TCCR

Para comenzar a utilizar este temporizador, lo más importante a tener en cuenta son los tres últimos bits del registro TCCRxB: CS12, CS11 y CS10. Que establecen las configuraciones del reloj. En la siguiente tabla se muestra la utilidad de cada configuración:

CS12	**CS11**	**CS10**	**Descripción**
0	0	0	Sin entrada para el reloj (Temporizador/Contador parado)
0	0	1	clk/1 (No prescaling)
0	1	0	clk/8 (From prescaler)
0	1	1	clk/64 (From prescaler)
1	0	0	clk/256 (From prescaler)
1	0	1	clk/1024 (From prescaler)
1	1	0	Fuente de reloj externa en el pin T5. Reloj en flanco de bajada.
1	1	1	Fuente de reloj externa en el pin T5. Reloj en flanco de subida.

Tabla 5.2: Descripción del CS (Clock Select Bit)

Por defecto, estos bits están establecidos a 0. Para calcular la palabra de cuenta se utiliza la siguiente formula:

$$tiempo = resolución\,timer \times (cuenta + 1)$$

$$(cuenta + 1) = tiempo \,/\, resolución\,timer$$

$$cuenta = \frac{tiempo}{resolución} + 1 = \frac{0{,}1s}{16{\cdot}10^6/256} + 1 = 11250 - 1 = 11249$$

Donde tiempo representa el periodo requerido (180 ms), la resolución del timer viene dada por la frecuencia del reloj interno (16 MHz) dividida por el divisor «prescaler» utilizado, en este caso se ha utilizado un divisor de 256.

Por último se debe incluir en la zona de declaración de librerías de la s-function las siguientes cabeceras:

```
#include <avr/io.h>
#include <avr/interrupt.h>
```

Y para configurar el temporizador con la palabra de cuenta previamente calculada se añade el siguiente código en la zona de inicialización de la s-function:

```
// Inicializa el timer 5.
cli();          // Desactiva las interrupciones globales.
TCCR5A = 0;     // Establece el registro TCCR5A a 0.
TCCR5B = 0;     // Establece el registro TCCR5B a 0.

// Establece la palabra de cuenta.
OCR5A = 11249;
// Habilita el modo CTC.
TCCR5B |= (1 << WGM12);
// Establece a 1 el bit CS12 para el prescaler 256.
TCCR5B |= (1 << CS12);
// Habilita la interrupción del timer.
TIMSK5 |= (1 << OCIE1A);
sei();          // Activa la interripciones globales.
```

Y también se añade el siguiente código en la pestaña «Output» para volcar el valor de la velocidad en las salidas del bloque simulink:

```
speed0[0]=Enc[0].speed;
speed1[9]=Enc[1].speed;
```

5.5.2. Bloque Sensor de Distancia

Se trata de un bloque que cuenta con una sola salida (la distancia en cm) y dos parámetros configurables correspondientes a los pines «echo» y «trigger» a los que está conectado el sensor, a continuación se muestra una figura con el bloque:

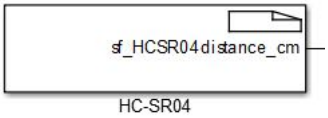


Figura 5.5.5: Bloque Sensor de Distancia

En este bloque se utiliza la librería «NewPing» [6]. Una librería que permite utilizar de forma trivial los siguientes modelos de sensores ultrasónicos: SR04, SRF05, SRF06, DYP-ME007 y Parallax. Entre sus características destacan las siguientes:

- Permite trabajar con resultados fiables a una velocidad de hasta 30 lecturas por segundo.
- Incorpora métodos de interrupción.
- Implementa un filtro de la mediana para la corrección de errores.
- Utiliza registros PORT [12] cuando accede a los pines; para una ejecución más rápida y para reducir el tamaño del código.
- Permite definir una distancia máxima.
- Facilita el uso de múltiples sensores de distancia.
- Permite obtener los resultados en diferentes unidades cm, pulgadas y uS.
- No utiliza «pulseIn», que es lento y no obtiene buenos resultados en algunos modelos de sensores ultrasónicos.

Para poder utilizar una librería externa en un bloque S-Function de Simulink, tal y como se describe en la sección 5.4, se incorpora en la pestaña de declaración de librerías la cabecera y el fichero de implementación correspondiente a la librería «NewPing»:

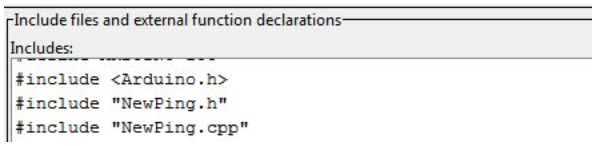


Figura 5.5.6: Librería «NewPing»

Estos dos ficheros deben encontrarse en la carpeta donde se encuentre el proyecto, siempre que se utilice este bloque, para que pueda ser compilado. Además en esta zona también se declara una instancia del objeto «NewPing»:

```
NewPing sonar(trigger[0],echo[0],200);
```

Donde «trigger» y «echo» son dos parámetros que dispone el bloque para definir los pines a los que se encuentra conectado el sensor:

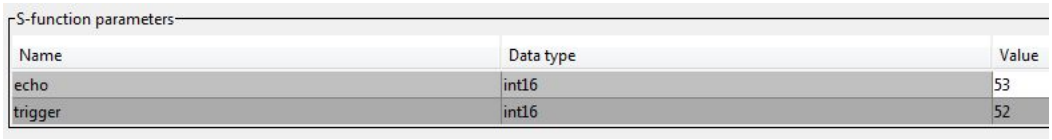


Figura 5.5.7: Parámetros Sensor Ultrasónico

El tercer parámetro (200) representa la distancia máxima en cm a partir de la cual se descartarán los pings enviados. Para actualizar el valor de la salida del bloque se añade el siguiente código en la pestaña «Output»:

```
unsigned int us = sonar.ping();
distance_cm[0] = us / US_ROUNDTRIP_CM;
```

donde la variable «distance_cm» es la salida del bloque, y el resultado utiliza la unidad cm.

Capítulo 6

Prácticas

En este capítulo se van a describir las dos prácticas desarrolladas en este proyecto. En la primera se llevara a cabo un control de posición y velocidad sobre los motores. Y en la segunda se desarrollará un sistema de navegación reactivo.

6.1. Práctica 1: Control de Posición y Velocidad (PID)

El objetivo principal de esta práctica consiste en realizar un control de posición y de velocidad de los motores EMG30 usando los encoders incorporados y un controlador PID en Simulink.

6.1.1. Objetivos de la Práctica

1. Realizar el control de posición de uno de los motores, para ello se deberá montar el esquema simulink necesario utilizando el bloque PID, para que el motor pueda ser posicionado en una posición de referencia.

2. En este segundo apartado se cambiará la referencia fija del apartado anterior por la posición del otro motor de la plataforma. El objetivo es que al girar un motor, el otro motor realice el mismo movimiento.

3. Realizar el control de velocidad para cada uno de los motores. Para ello se deberá montar el esquema simulink necesario usando el bloque PID de simulink para que el motor pueda seguir una referencia fija en rpm. Se debería intentar ajustar los dos PIDs para que ambos motores tuvieran una respuesta similar.

6.1.2. Requisitos

- Ordenador con el software MATLAB/Simulink
- Librería de Arduino de Simulink.
- Bloque DualQuadEncoder proporcionado (se proporcionará un fichero comprimido pr1.zip cuyo contenido deberá ser extraído en el path donde se encuentre el esquema Simulink de la práctica).
- Plataforma PIERO con los siguiente elementos:
  - 2 Motores EMG30 con soportes y ruedas.
  - Arduino Mega con la shield de conexiones.
  - Controlador Dual de motor L298N.
  - Batería y cables de conexión tipo dupont.

6.1.3. Esquema de Conexiones

En la siguiente figura podemos observar un esquema de las conexiones físicas entre los diferentes componentes utilizados en esta práctica.

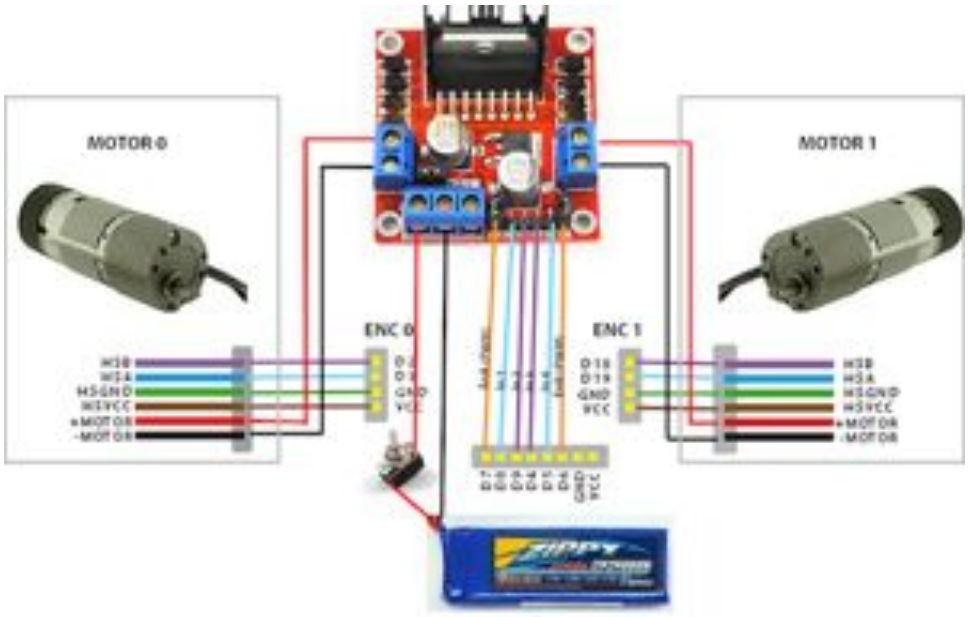


Figura 6.1.1: Esquema de Conexiones

Destacar que los encoders de ambos motores están conectados a los puertos de la shield etiquetados como ENC0 y ENC1. Éstos están conectados a las entradas digitales 2,3,18 y 19, que son entradas para las que en el Arduino Mega tiene asociadas interrupciones reales.

El controlador dual de motores esta conectado al puerto de la shield etiquetado como L298N. En la siguiente tabla se muestran las conexiones y el sentido de giro de los motores en función del valor de las entradas *INX*.

	L298N	**Arduino**	**Adelante**	**Atrás**
ENC 0	ENA	D7	PWM	PWM
	IN1	D8	1	0
	IN2	D9	0	1
ENC 1	ENB	D6	PWM	PWM
	IN3	D4	0	1
	IN4	D5	1	0

Tabla 6.1: Conexiones de los Motores

6.1.4. Guía de Realización

La primera tarea a llevar a cabo será obtener el valor PWM mínimo que provoca que el motor comience a moverse, este valor lo utilizaremos posteriormente como offset en la salida del PID. Para ello debemos montar en simulink un esquema básico, como el que podemos observar en la siguiente figura:

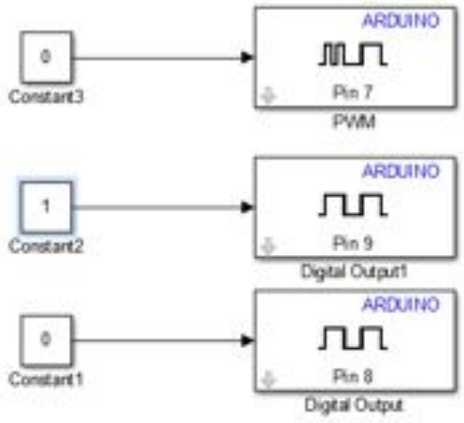


Figura 6.1.2: Esquema Básico Motores

Ejecutamos el modelo en el Arduino en modo external. Vamos incrementando el valor de la constante que le pasamos como PWM a las entradas ENA y ENB hasta que estos comiencen a girar.

Una vez obtenido el offset de los motores se procede a montar el esquema básico de la práctica que puede verse en la siguiente figura:

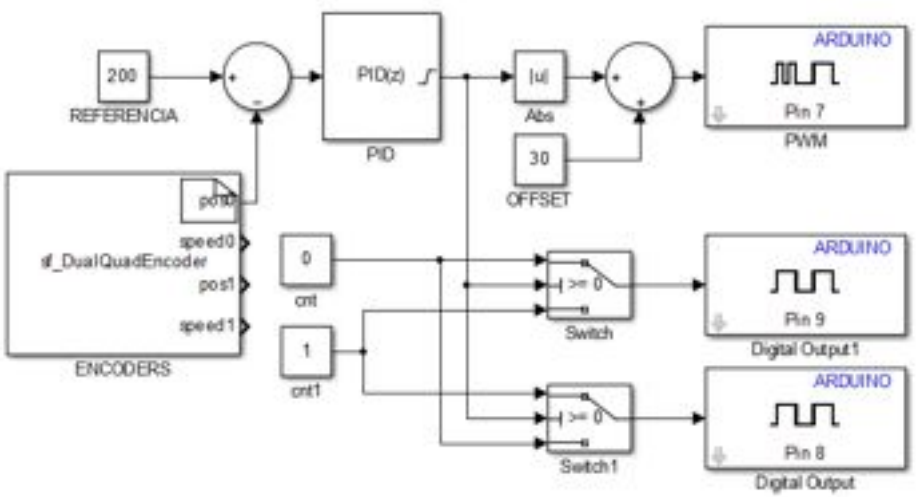


Figura 6.1.3: Esquema Simulink PID posición

En ella se puede observar como se está utilizando el bloque PID que proporciona Simulink. La entrada del bloque es el error de posición, que se obtiene restando a la referencia el valor de posición obtenido del bloque DualQuadEncoder. El signo de la salida del PID se utiliza para seleccionar el sentido de giro del motor (véase la tabla 6.1.3). Y como PWM se le pasa el valor absoluto de la salida del bloque PID al que se le suma el offset calculado previamente. En las pruebas que se han desarrollado para este proyecto el valor mínimo necesario era 35, pero se utiliza un valor un poco inferior a este offset.

A continuación se pasa a configurar el PID, para ello se abren las preferencias del bloque. En la pestaña principal como observamos en la siguiente figura:

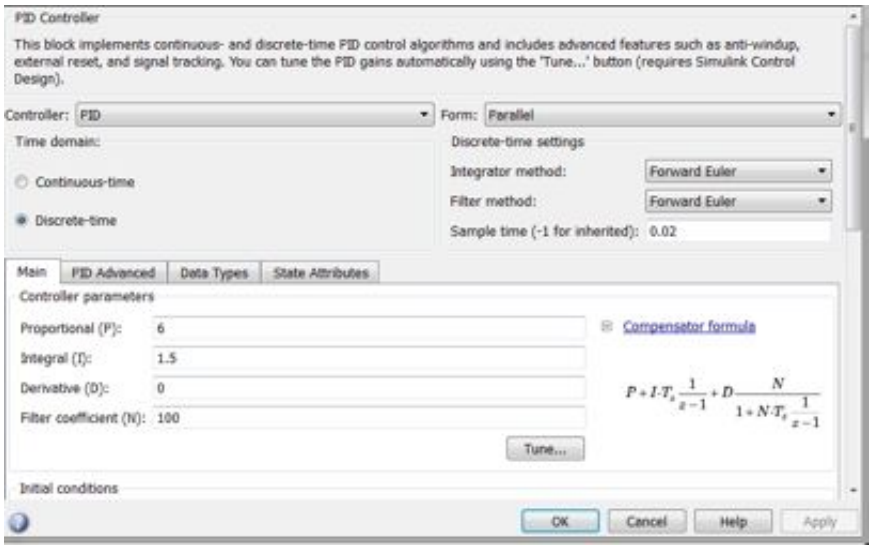


Figura 6.1.4: Preferencias PID

se selecciona la forma del PID, en este caso paralelo aunque también es posible seleccionar el ideal, también el tipo; de tiempo discreto con periodo 0.02 segundos. En la pestaña main es donde se deben modificar cada una de las componentes del controlador, la proporcional (P), la integral (I) y la derivativa (D). La pestaña avanzada ofrece una opción bastante interesante, se trata de un sistema «antiwindup» y un sistema de saturación para limitar los valores máximos y mínimos de la salida. A la hora de configurar estos limites hay que tener en cuenta que a la salida se le está sumando un offset de 30. En el caso del control de posición dado que no se requieren velocidades muy altas, estos límites se han establecido en 100 y -100, que al sumarle posteriormente el offset quedaran en 130 y -130.

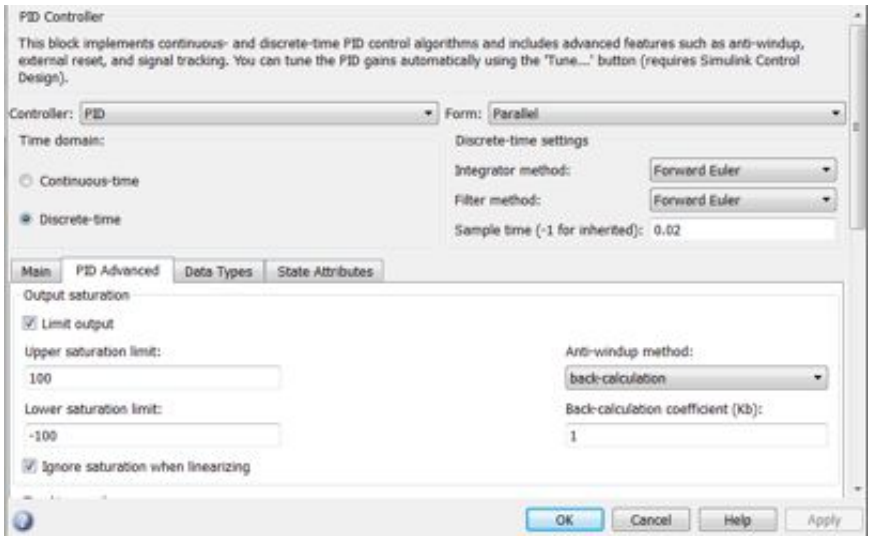


Figura 6.1.5: Preferencias PID Avanzado

Una vez montado el esquema básico se pasa a sintonizar el PID mediante el método de Ziegler–Nichols[7] basada en la ganancia crítica K_{cr}, y en el periodo crítico P_{cr}. En la siguiente tabla se puede observar los parámetros de sintonización del PID de Ziegler–Nichols:

Controlador	K_p	T_i	T_d
P	$0{,}5\,K_{cr}$	∞	0
PI	$0{,}45\,K_{cr}$	$\frac{1}{1{,}2}\,P_{cr}$	0
PID	$0{,}6\,K_{cr}$	$0{,}5\,P_{cr}$	$0{,}125\,P_{cr}$

Tabla 6.2: Regla de Ziegler-Nichols

El siguiente punto es obtener la ganancia crítica K_{cr} y el periodo crítico P_{cr}. Para ello se selecciona el PID paralelo, se establecen las componentes I y D a 0, y se van incrementando el valor de la ganancia proporcional P hasta que el motor entre en oscilación permanente. En ese momento se mide el periodo de oscilación P_{cr}. Para realizar esta medida se debe utilizar el scope junto con el modo external o depuración que nos ofrece Simulink a la hora de ejecutar el bloque en el Arduino.

Se pueden obtener mediante la tabla los valores de las ganancias P, I y D, pero hay que tener en cuenta el tipo de PID seleccionado en el bloque PID de Simulink. Dado que en la tabla anterior los parámetros hacen referencia al PID estándar:

$$G\left(s\right) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

y en el PID paralelo:

$$G\left(s\right) = P + I\frac{1}{s} + Ds$$

que es el que se había mostrado previamente en la figura anterior los parámetros son diferentes. La conversión de parámetros es trivial:

$$P = K_p$$
$$I = \frac{K_p}{T_i}$$
$$D = K_p T_d$$

Una vez calculados los diferentes parámetros, se insertan los valores en el bloque PID y se carga el código en el Arduino. Se comprueba el resultado utilizando el external mode y las gráficas scope. Para ajustar el PID manualmente podemos utilizar la siguiente tabla en la que vemos los efectos en la respuesta del PID ante las diferentes modificaciones en los parámetros:

	T. de Subida	Sobreoscilación	T. de Est.	Error R. Permanente
P	Decrementa	Incrementa	Cambio Pequeño	Decrementa
I	Decrementa	Incrementa	Incrementa	Elimina
D	Cambio Pequeño	Decrementa	Decrementa	Cambio Pequeño

Tabla 6.3: Parámetros PID

En las pruebas realizadas durante el desarrollo del presente proyecto se ajustó el PID, obteniendo como resultado las siguientes ganancias:

$$\begin{cases} P = 4 \\ I = 2 \\ D = 0 \end{cases}$$

En las siguientes gráficas se puede observar la respuesta del controlador PID diseñado ante modificaciones en la referencia, donde la gráfica superior representa la señal de referencia y la inferior la posición del motor:

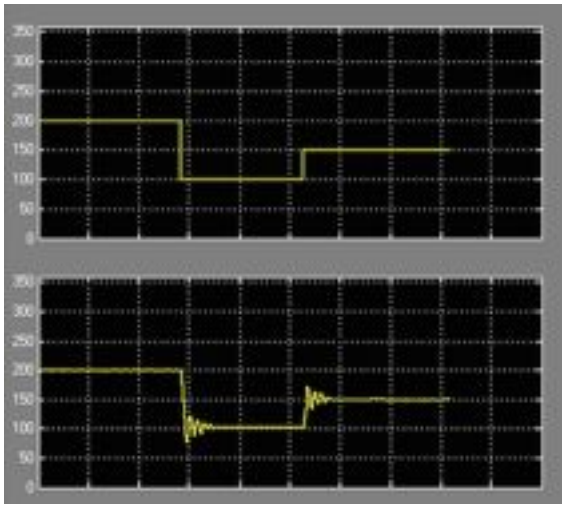


Figura 6.1.6: Respuesta PID Posición

Por lo tanto en este punto se habría completado del apartado 1 de la presente práctica. El apartado 2 es relativamente trivial pues el único cambio a realizar es eliminar la constante utilizada como referencia, y conectar la referencia con la salida pos1 del bloque DualQuadEncoder. Obteniendo como resultado un esquema similar al que se muestra en la siguiente gráfica:

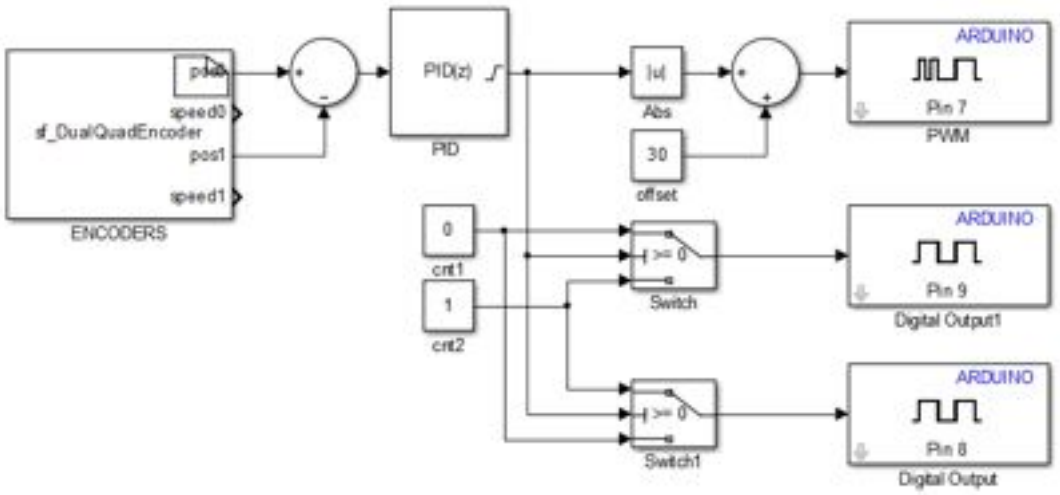


Figura 6.1.7: PID posición Apartado 2

Y para probar el resultado se deberá cargar el esquema Simulink en el Arduino, y al mover una de las ruedas se comprueba como la otra rueda sigue su movimiento. En la siguiente gráfica se observa este resultado, la gráfica superior representa la posición de la rueda de referencia y la inferior la posición de la rueda controlada:

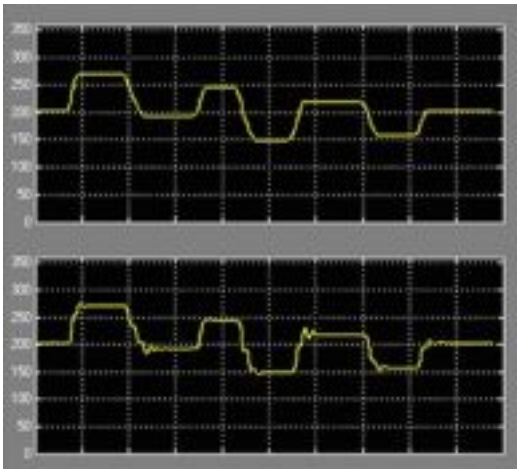


Figura 6.1.8: Resultado Apartado 2

El siguiente paso a realizar es el control PID de velocidad. Para ello se deben seguir los mismos pasos realizados para sintonizar el PID de posición. La principales modificaciones a realizar son el cambio de la posición por la velocidad, obteniendo un esquema como el siguiente:

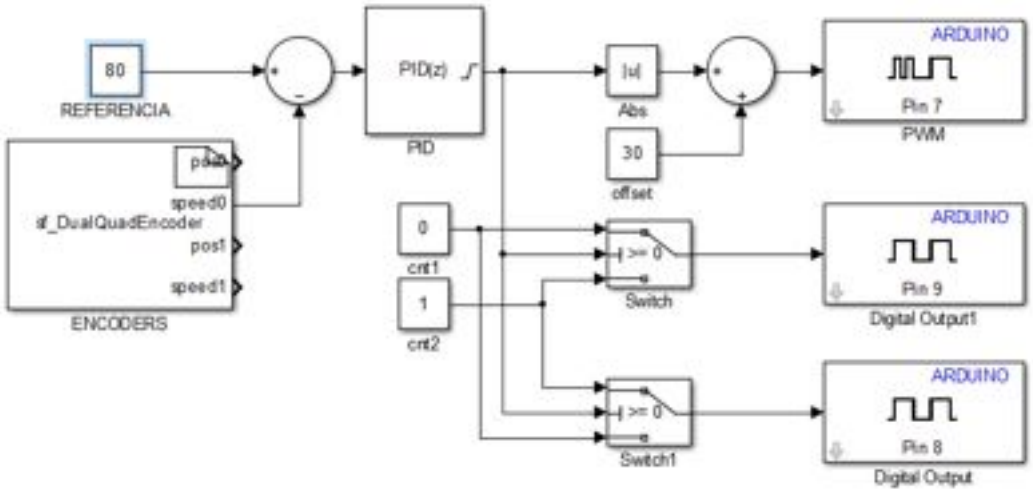


Figura 6.1.9: Esquema Control de Velocidad

Otro cambio importante que se debe tener en cuenta son los límites de saturación seleccionados para la salida del PID. En este caso se establecerán los valores 225 y -225. Dado que la señal PWM máxima es 255, a la que le restamos el offset establecido de 30.

En la pruebas realizadas en el presente proyecto se establecieron las siguientes ganancias para el control de ambos motores:

	P	**I**	**D**
Motor 0	0.9	0.75	0
Motor 1	0.8	0.75	0

Tabla 6.4: Ganancias Control de Velocidad

Y en las siguientes gráficas se pueden observar las respuestas de ambos motores ante entradas escalón en la referencia. A la derecha el motor 0 y a la izquierda el motor 1:

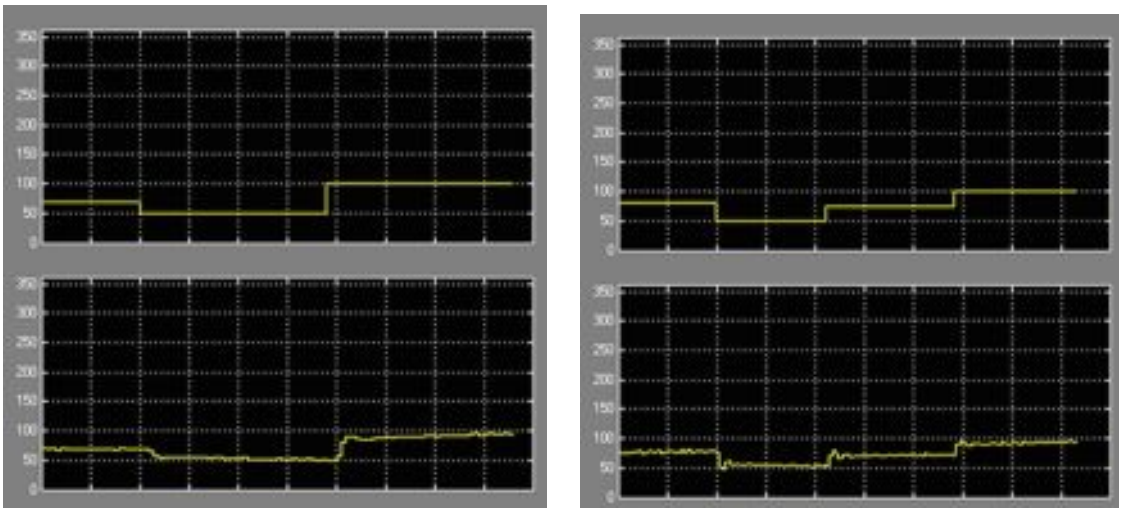


Figura 6.1.10: PID Velocidad

Y como recomendaciones finales para intentar perder el menor tiempo posible en la realización de las prácticas, se recomienda asegurarse de que los tornillos de las ruedas están convenientemente ajustados antes de realizar cualquier prueba sobre ellos. También es importante que se utilicen los mismos tiempos de sampleos en todos los bloques, dado que esto puede provocar resultados muy extraños a la hora de ajustar las ganancias de los controladores PID.

6.2. Práctica 2: Sistema de Navegación Reactivo

Realizar un sistema de navegación reactivo en Simulink utilizando la plataforma PIERO, el sensor de distancias y módulo bluetooth para modificar la velocidad de referencia.

6.2.1. Objetivos de la Práctica

1. Conseguir que el sistema de avance hasta que encuentre un obstáculo. Para detectar el obstáculo se utilizará el sensor de distancias ultrasónico HC-SR04. Una vez encontrado el obstáculo la plataforma deberá girar hasta que no exista ningún obstáculo delante de la misma, y posteriormente seguir avanzando.

2. La velocidad de ambas ruedas deberá ser constante y controlada por dos controladores PID, se debería utilizar los mismos controladores implementados en la Práctica 1.

3. Esta velocidad podrá ser cambiada online, para ello se deberá implementar un sistema que permita modificar la velocidad de referencia utilizando el sensor de bluetooth.

6.2.2. Requisitos

- Ordenador con el software MATLAB/Simulink
- Librería de Arduino de Simulink.
- Software de emulación de terminal serie sobre bluetooth (Por ejemplo la aplicación «Bluetooth Terminal»[8] para Android o «Putty» para Windows).
- Bloque DualQuadEncoder proporcionado (se proporcionará un fichero comprimido pr1.zip cuyo contenido deberá ser extraído en el path donde se encuentre el esquema Simulink de la práctica).
- Bloque HC-SR04 proporcionado.
- Plataforma PIERO con los siguiente elementos:
  - 2 Motores EMG30 con soportes y ruedas.
  - Arduino Mega con la shield de conexiones.
  - Controlador Dual de motor L298N.
  - Sensor de distancias HC-SR04.
  - Módulo de bluetooth.
  - Batería y cables de conexión tipo dupont.

6.2.3. Esquema de Conexiones

En la siguiente figura podemos observar un esquema de las conexiones físicas entre los diferentes componentes utilizados en esta práctica.

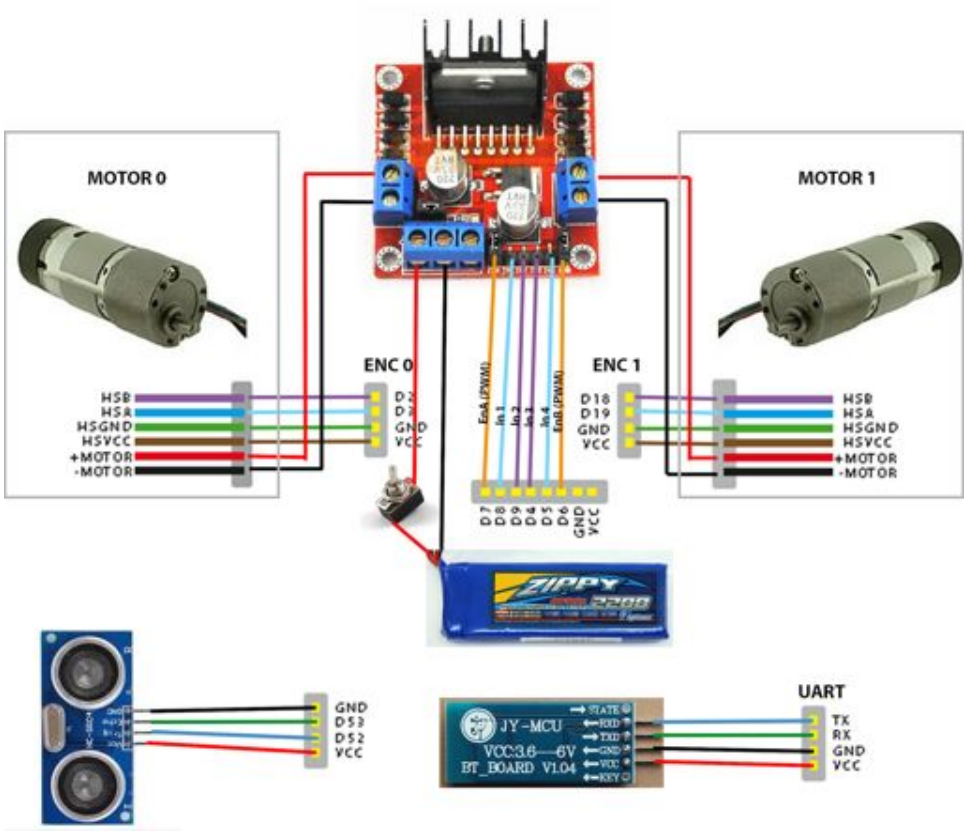


Figura 6.2.1: Esquema de Conexiones

6.2.4. Guía de Realización

Como primer paso se añade un bloque StateFlow que tiene como objetivo la creación de un diagrama de flujo lógico de manera muy sencilla. Para esta práctica se han estimado necesarios dos estados, el estado de avanzar y el estado de girar. Y la transición de un estado a otro viene dada por la distancia obtenida por el sensor de distancias ultrasónico. En la siguiente gráfica podemos observar este diagrama creado con la toolbox StateFlow de Simulink:

Figura 6.2.2: Diagrama de Estados

El estado inicial viene indicado por la flecha. Y la condición para girar la plataforma es la detección de un objeto a una distancia inferior o igual a 15 cm. En el interior de cada estado modificamos el valor de las salidas del bloque. Dicho bloque dispone de dos salidas y una entrada. La entrada es «dist» y representa la distancia. Y las salidas «vel_izq» y «vel_der», representan el sentido de giro de cada rueda (1 avanzar y -1 retroceder).

Una vez creado este sistema se puede montar el diagrama de bloques general, obteniendo algo similar a lo que se muestra a continuación en la siguiente figura:

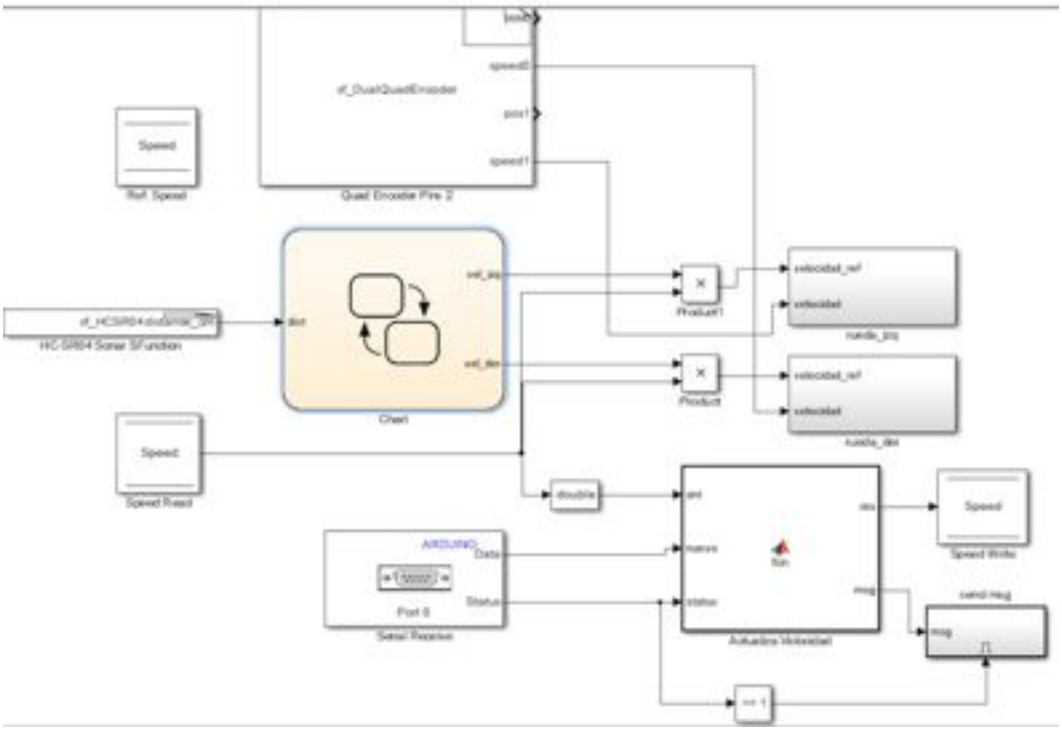


Figura 6.2.3: Diagrama de Bloques Simulink

En este diagrama se encuentra el bloque utilizado en las Práctica 1 DualQuadEncoder, que permite obtener las velocidades de ambos motores. Esta velocidad es multiplicada por la

salida del bloque «chart», que contiene el diagrama de estados descrito previamente. De esta manera se obtiene el signo o sentido de la velocidad, y esta velocidad junto con la velocidad de referencia es la que se utiliza en los bloques «rueda_izq» y «rueda_der» que contienen los controladores PID diseñados en la práctica anterior. En este diagrama se ha incluido el bloque HCSR04 que permite obtener la distancia en centímetros procedente del sensor ultrasónico HCSR04.

El resto de bloques se encargan de la gestión de la velocidad de referencia. Para guardar dicha velocidad, se utiliza un bloque «DataStoreMemory», que permite leer y modificar el valor de la variable (speed) contenida en este bloque. Inicialmente se establece una velocidad de referencia fija, se puede modificar en las preferencias del bloque Ref. Speed, como se puede observar en la siguiente figura, donde dicho valor se ha establecido en 80 rpm:

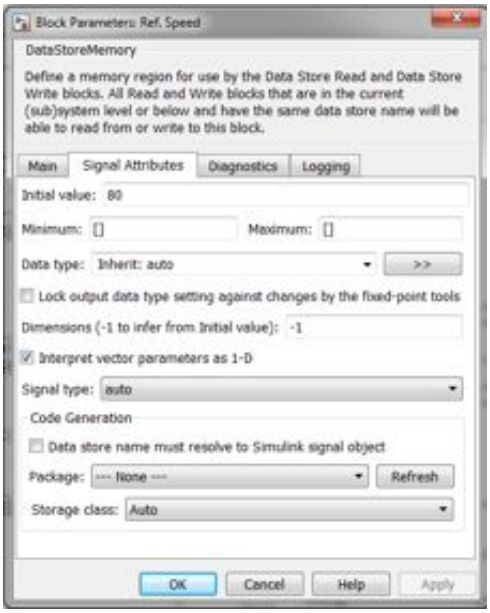

Block Parameters: Ref. Speed
DataStoreMemory
Define a memory region for use by the Data Store Read and Data Store Write blocks. All Read and Write blocks that are in the current (sub)system level or below and have the same data store name will be able to read from or write to this block.
Main
Signal Attributes
Diagnostics
Logging
Initial value: 80
Minimum: []
Maximum: []
Data type: Inherit: auto
>>
Lock output data type setting against changes by the fixed-point tools
Dimensions (-1 to infer from Initial value): -1
Interpret vector parameters as 1-D
Signal type: auto
Code Generation
Data store name must resolve to Simulink signal object
Package: --- None ---
Refresh
Storage class: Auto
OK
Cancel
Help
Apply

Figura 6.2.4: DataStoreMemory

Por otra parte, el bloque «Actualiza Velocidad» se encarga en cada ciclo de comprobar si existe un dato nuevo recibido por el puerto serie especificado, aquél donde esté conectado el modulo bluetooth. Lo que realiza este bloque es lo siguiente:

- En el caso de no recibir ningún dato por el puerto serie (status es igual a 0) devuelve en la salida «res» el mismo valor de velocidad anterior y éste se escribe en la variable speed.
- En el caso de recibir el caracter «V», se mantiene la velocidad actual y en la salida «msg» se devuelve el valor de la velocidad de referencia actual en un mensaje de texto con el siguiente formato (donde «X» es la velocidad de referencia actual):

`recibo: V. vel: «X»`

- En el caso de recibir un caracter númerico se almacena dicho número en un buffer de tamaño 3, que será utilizado para modificar la velocidad. Si el buffer esta completo se ignora el caracter recibido y si no está completo se devuelve el siguiente mensaje:

  `recibo: «X». guard.`

- Si se recibe el caracter «A», se comprueba si el buffer contiene una serie de caracteres que representen una velocidad aceptable, un número contenido en el rango 0-170. Se modifica la velocidad de referencia y además se devuelve el siguiente mensaje:

  `recibo: A. vel: «X»`

- En el caso de recibir cualquier otro carácter, no se realizará ninguna acción especial. Tan solo se devolverá un mensaje con el siguiente formato (donde «C» representa el caracter recibido):

  `recibo: «C»`

El bloque «send msg» es el encargado de enviar vía bluetooth el mensaje de respuesta. Dispone de un trigger, por lo tanto solo se va a ejecutar cuando el valor de status sea igual a 1, es decir cuando se reciba un dato por el puerto de serie. El contenido de dicho bloque se puede observar en la siguiente figura:

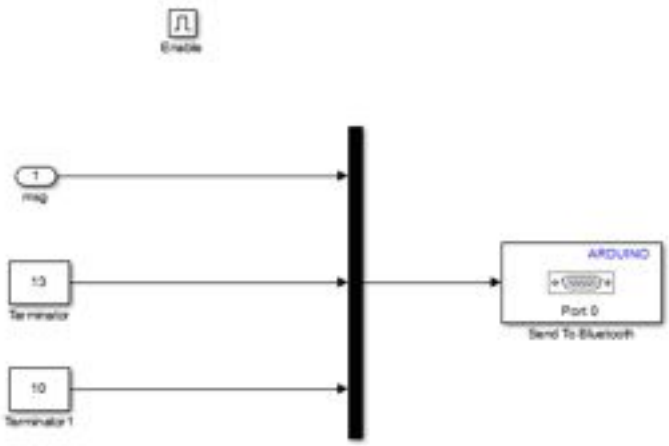


Figura 6.2.5: Bloque «send msg»

Al final del mensaje se le concatenan los caracteres 13 y 10, en la tabla ASCII representan LF (nueva linea) y CR (retorno de carro).

De esta manera queda completada la tarea que se requería en esta práctica. Para testearla se carga el diagrama de bloques en el Arduino, pero esta vez no en modo external. Y una vez cargado se puede comprobar cómo, si los dos PIDs están ajustados correctamente, la plataforma realizará una trayectoria recta, cuando no detecte ningún obstáculo. Cuando los dos PIDs tienen tiempos de establecimientos distintos se podrá comprobar que la plataforma girará un poco hasta que ambos PIDs hayan alcanzado el estado de régimen permanente.

Y para poder modificar la velocidad de referencia, tan solo hay que conectar el emulador de terminal bluetooth con el Arduino y enviar el siguiente texto:

```
50A
```

De esta manera se estará modificando la velocidad de referencia a 50 rpm.

Capítulo 7

Conclusiones y Trabajos Futuros

Como conclusiones finales destacar que se han cumplido los objetivos propuestos en el presente proyecto, diseñando y desarrollando una plataforma de bajo coste, por un precio inferior a 200€ por unidad como se puede comprobar en el presupuesto adjunto.

En cuanto al desarrollo de prácticas con Simulink, aunque la librería inicial ofrecida por Mathwork se puede considerar un poco escasa, gracias al uso de los bloques S-Function se pueden ampliar y desarrollar cualquier tipo de función que se puede realizar en el lenguaje propio de Arduino, al igual que el uso de librerías Arduino escritas en c++ y altamente testeadas.

Se han desarrollado prácticas para llevar a cabo el control de los motores tanto de posición como de velocidad. Los resultados obtenidos, a pesar de la baja resolución de los encoders de los motores, son bastante notables, esto se puede comprobar en el capítulo 6, . También se ha utilizado la toolbox StateFlow de Simulink en la segunda práctica, comprobando su correcto funcionamiento en la plataforma Arduino. Esta toolbox facilita al usuario/alumno la programación de tareas de control basado en sistemas de eventos discretos o de diagnósticos de fallos.

Por otra parte uno de los objetivos del proyecto consistía en la creación de un sistema para facilitar la conexión de los diferentes sensores/actuadores, con la desarrollo de la shield de conxiones se cumple con creces este objetivo. Esta shield de conexiones a diferencia de las shield habituales que proporcionan infinidad de conexiones de tres hilos (GND, VCC y señal), dado que los dispositivos interesantes suelen requerir dos lineas de señal: módulos de comunicaciones, sensores I2C, codificadores incrementales, etc. ofrece 4. Por ello se ha adoptado un sistema de conexiones basado en la shield GROVE [14], que usa estas conexiones de 4 hilos para todas las entradas y salidas para el desarrollo de la shield de este proyecto.

7.1. Trabajos Futuros

A pesar de que en este proyecto se ha utilizado el módulo de bluetooth en la plataforma para enviar y recibir datos, el uso realmente interesante para este módulo sería como sustituto del cable usb, tanto para la carga del programa desde Simulink, como para la depuración en tiempo real usando el modo externo. Por lo tanto esta sería una mejora bastante interanse para la plataforma que facilitaría la tarea de la depuración de los programas diseñados por los alumnos al no tener que estar conectados mediante un cable USB al pc.

Además una posible mejora para la plataforma podría un indicador luminoso de la carga de la batería, de esta manera el profesor o responsable del laboratorio dispondría de un sistema fácil y rápido para comprobar el estado de carga de las mismas y poder llevar a cabo la carga antes de que las clases prácticas tengan lugar.

Y evidentemete un trabajo siempre abierto sería el aumento del número de prácticas utilizando diversos sensores (girospocio, sensores siguelineas...), que en conjunto con la toolbox StateFlow de Simulink crearían prácticas interesantes de cara a la docencia, atractivas de cara al alumno y con un coste temporal de desarrollo bajo, lo que permite que puedan llevarse a cabo en una sesión de laboratorio.

Bibliografía

[1] Libreria de Componentes de SparkFun. https://github.com/sparkfun/sparkfun-eagle-libraries.

[2] Libreria Arduino en Simulink. http://www.mathworks.com/help/simulink/arduino.html.

[3] Paquete ARDUINO Support from Simulink. http://www.mathworks.com/hardware-support/arduino-simulink.html.

[4] J.M. Gabriel. Piero mobile robot platform. http://gomezdegabriel.com/wordpress/projects/piero-mobile-robot-platform/. 2014.

[5] K. Lamar and A. G. Kocsis. Implementation of speed measurement for electrical drives equipped with quadrature encoder in labview fpga. *Acta Technica Corviniensis - Bulletin of Engineering*, 2013.

[6] Libreria Arduino NewPing. http://playground.arduino.cc/code/newping.

[7] Katsushiko Ogata. *Ingenieria de Control Moderna (Spanish Edition)*. Prentice Hall, 1999.

[8] App para Android Bluetooth Terminal. https://play.google.com/store/apps/details?id=qwerty.bluetoothterminal.

[9] Manual para creacion de drivers en Simulink. http://www.mathworks.com/matlabcentral/fileexchange/39354-device-drivers.

[10] Raspberry pi Quadrature Encoder Sfuntion. http://www.mathworks.com/matlabcentral/fileexchange/41645-raspberry-pi-quadrature-encoder-sfunction.

[11] Documentacion Arduino PinMode. http://arduino.cc/en/reference/pinmode.

[12] Documentacion Arduino PortManipulation. http://www.arduino.cc/es/reference /port-manipulation.

[13] Documentacion Mathwork S-Function-Builder. http://www.mathworks.se/help/simulink/slref/sfunctionbuilder.html.

[14] Seeed Studio. Grove mega shield. http://www.seeedstudio.com/wiki/grove_-_mega_shield.

II

PRESUPUESTO

1. Plataforma móvil

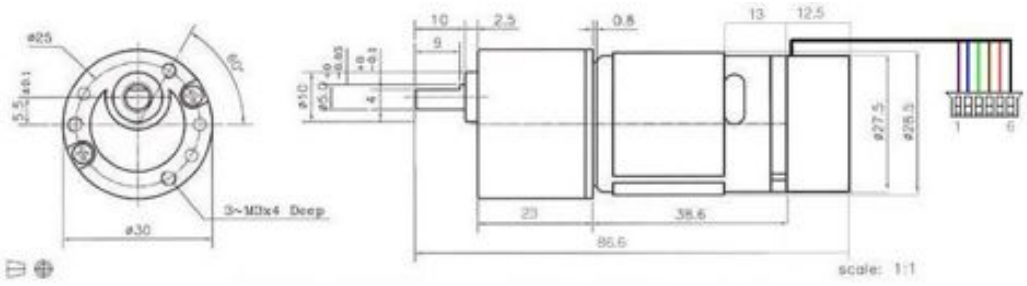
DESCRIPCIÓN	UD.	PRECIO UD.	PRECIO
Arduino Mega	1	21.75 €	21.75 €
Plataforma Base	1	5 €	5 €
PeroShield	1	4.80 €	4.80 €
L298N Dual H Bridge	1	2.44 €	2.44 €
Motor EMG30	2	37.35 €	74.70 €
Ruedas Motores	2	12.25 €	24.50 €
Soporte Motores	2	4.35 €	8.70 €
Batería LI-PO 2200 mah	1	8.76 €	8.76 €
Interruptor	1	0.10 €	0.10 €
Tiras de pines machos	10	0.18 €	1.80 €
Conectores hembra	2	1.20 €	2.40 €
Conectores dobles hembra	2	0.60 €	1.20 €
Bluetooth Connector	1	13.65 €	13.65 €
Tornillos de Nylon y Sep. (M2.5)	10	0.05 €	0.50 €
Sensor Ultrasónico	1	1.79 €	1.79 €
Conector XT60	1	0.5 €	0.5 €
Conector DC 2.1 mm	1	1.65 €	1.65 €
Cableado	1	3.00 €	3.00 €
Placa Soporte Batería	2	0.30 €	0.60 €
Escuadra Soporte Batería	2	0.30 €	0.60 €
		TOTAL:	178,44 €

III

ANEXOS

ANEXO A - Especificaciones para los Motores EMG30 y para los Soportes.

Measurements



Connector

The EMG30 is supplied with a 6 way JST connector (part no PHR-6) at the end of approx 90mm of cable as standard. The connections are:

Wire colour	Connection
Purple (1)	Hall Sensor B Vout
Blue (2)	Hall Sensor A Vout
Green (3)	Hall sensor ground
Brown (4)	Hall sensor Vcc
Red (5)	+ Motor
Black (6)	- Motor

Wire colours are from the actual cable.

The hall sensors accept voltages between 3.5v and 20v.

The outputs are open collector and require pull-ups to whatever signal level is required. On the MD25 they are powered from 12v and pulled up to 5v for the signals.

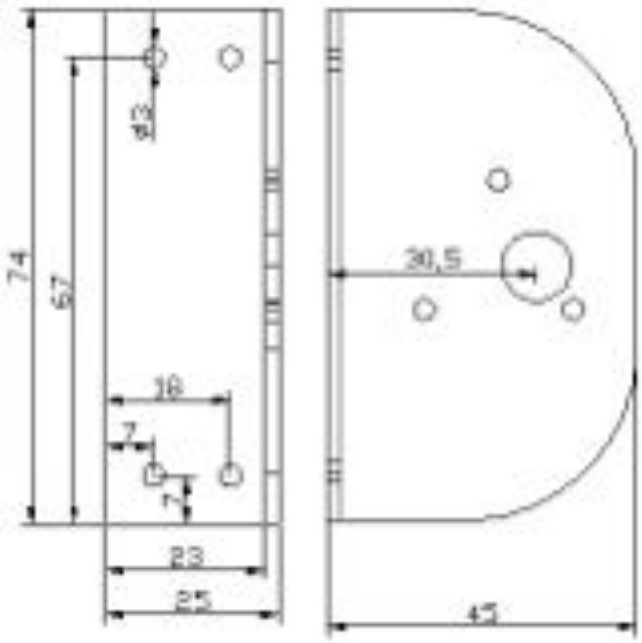
Specification

Rated voltage	12v
Rated torque	1.5kg/cm
Rated speed	170rpm
Rated current	530mA
No load speed	216
No load current	150mA
Stall Current	2.5A
Rated output	4.22W
Encoder counts per output shaft turn	360

Measured Shaft Speed when used off-load with MD23 and 12v supply

Minimum Speed	1.5rpm
Maximum Speed	200rpm

EMG30 Mounting Bracket



ANEXO B - HC-SR04 Manual de Usuario

User's Manual

V1.0

May 2013

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Cytron Technologies Incorporated with respect to the accuracy or use of such information or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Cytron Technologies's products as critical components in life support systems is not authorized except with express written approval by Cytron Technologies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Index

1. Introduction 3

2. Packing List 4

3. Product Layout 5

4. Product Specification and Limitation 6

5. Operation 7

6. Hardware Interface 8

7. Example Code 9

8. Warranty 10

1.0 INTRODUCTION

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats or dolphins do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1" to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

Features:

- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Currnt: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

Created by Cytron Technologies Sdn. Bhd. – All Rights Reserved

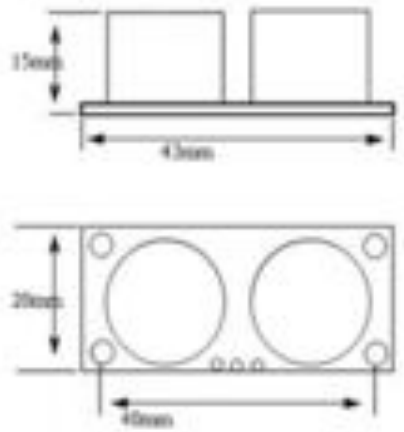
2.0 PACKING LIST

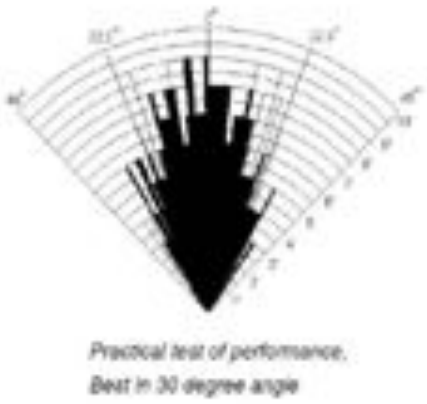
1. 1 x HC-SR04 module

Created by Cytron Technologies Sdn. Bhd. – All Rights Reserved

3.0 PRODUCT LAYOUT

VCC = +5VDC
Trig = Trigger input of Sensor
Echo = Echo output of Sensor
GND = GND





4.0 PRODUCT SPECIFICATION AND LIMITATIONS

Parameter	Min	Typ.	Max	Unit
Operating Voltage	4.50	5.0	5.5	V
Quiescent Current	1.5	2	2.5	mA
Working Current	10	15	20	mA
Ultrasonic Frequency	-	40	-	kHz

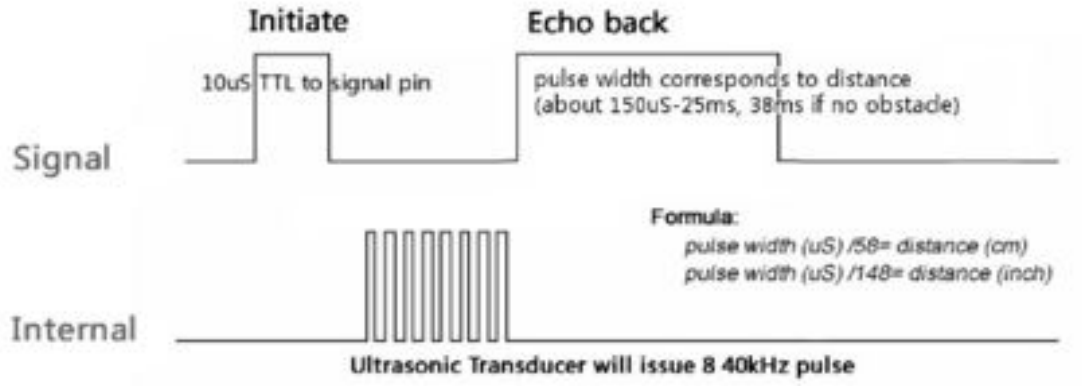
Created by Cytron Technologies Sdn. Bhd. – All Rights Reserved

5.0 OPERATION

The timing diagram of HC-SR04 is shown. To start measurement, Trig of SR04 must receive a pulse of high (5V) for at least 10us, this will initiate the sensor will transmit out 8 cycle of ultrasonic burst at 40kHz and wait for the reflected ultrasonic burst. When the sensor detected ultrasonic from receiver, it will set the Echo pin to high (5V) and delay for a period (width) which proportion to distance. To obtain the distance, measure the width (Ton) of Echo pin.

Time = Width of Echo pulse, in uS (micro second)

- Distance in centimeters = Time / 58
- Distance in inches = Time / 148
- Or you can utilize the speed of sound, which is 340m/s

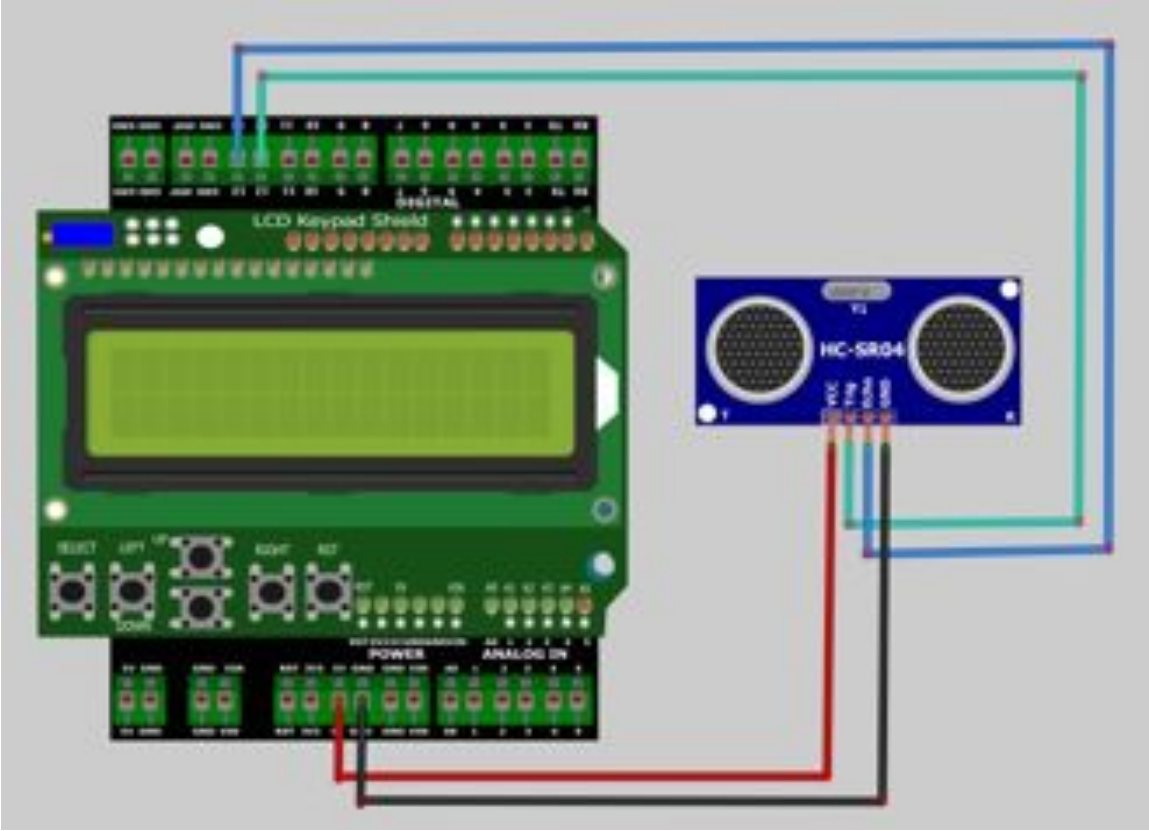


Note:

- Please connect the GND pin first before supplying power to VCC.
- Please make sure the surface of object to be detect should have at least 0.5 meter^2 better performance.

6.0 HARDWARE INTERFACE

Here is example connection for Ultrasonic Ranging module to Arduino UNO board. It can be interface with any microcontroller with digital input such as PIC, SK40C, SK28A, SKds40A, Arduino series.



7.0 EXAMPLE CODE

This is example code Ultrasonic Ranging module. Please download the complete code at the product page.

```
#include "Ultrasonic.h"
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
Ultrasonic ultrasonic(12,13);

void setup() {
lcd.begin(16, 2);
lcd.setCursor(0, 0);
lcd.print("HC-SR4 testing..");
delay(1000);

}

void loop()
{
  //lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print(ultrasonic.Ranging(CM));
  lcd.print("cm  ");

  delay(100);
}
```

8.0 WARRANTY

- Product warranty is valid for 6 months.
- Warranty only applies to manufacturing defect.
- Damaged caused by miss-use is not covered under warranty
- Warranty does not cover freight cost for both ways.

Prepared by
Cytron Technologies Sdn. Bhd.
19, Jalan Kebudayaan 1A,
Taman Universiti,
81300 Skudai,
Johor, Malaysia.

Tel: *+607-521 3178*
Fax: *+607-521 1861*

URL: www.cytron.com.my
Email: support@cytron.com.my
sales@cytron.com.my

Created by Cytron Technologies Sdn. Bhd. – All Rights Reserved

ANEXO C - HC-06 Manual de Usuario

HC Serial Bluetooth Products

User Instructional Manual

1 Introduction

HC serial Bluetooth products consist of Bluetooth serial interface module and Bluetooth adapter, such as:

(1) Bluetooth serial interface module:

Industrial level: HC-03, HC-04(HC-04-M, HC-04-S)

Civil level: HC-05, HC-06(HC-06-M, HC-06-S)

HC-05-D, HC-06-D (with baseboard, for test and evaluation)

(2) Bluetooth adapter:

HC-M4

HC-M6

This document mainly introduces Bluetooth serial module. Bluetooth serial module is used for converting serial port to Bluetooth. These modules have two modes: master and slaver device. The device named after even number is defined to be master or slaver when out of factory and can't be changed to the other mode. But for the device named after odd number, users can set the work mode (master or slaver) of the device by AT commands.

HC-04 specifically includes:

Master device: HC-04-M, M=master

Slave device: HC-04-S, S=slaver

The default situation of HC-04 is slave mode. If you need master mode, please state it clearly or place an order for HC-O4-M directly.The naming rule of HC-06 is same.

When HC-03 and HC-05 are out of factory, one part of parameters are set for activating the device. The work mode is not set, since user can set the mode of HC-03, HC-05 as they want.

The main function of Bluetooth serial module is replacing the serial port line, such as:

1. There are two MCUs want to communicate with each other. One connects to Bluetooth master device while the other one connects to slave device. Their connection can be built once the pair is made. This Bluetooth connection is equivalently liked to a serial port line connection including RXD, TXD

signals. And they can use the Bluetooth serial module to communicate with each other.

2. When MCU has Bluetooth salve module, it can communicate with Bluetooth adapter of computers and smart phones. Then there is a virtual communicable serial port line between MCU and computer or smart phone.

3. The Bluetooth devices in the market mostly are salve devices, such as Bluetooth printer, Bluetooth GPS. So, we can use master module to make pair and communicate with them.

Bluetooth Serial module's operation doesn't need drive, and can communicate with the other Bluetooth device who has the serial. But communication between two Bluetooth modules requires at least two conditions:

(1) The communication must be between master and slave.

(2) The password must be correct.

However, the two conditions are not sufficient conditions. There are also some other conditions basing on different device model. Detailed information is provided in the following chapters.

In the following chapters, we will repeatedly refer to Linvor's (Formerly known as Guangzhou HC Information Technology Co., Ltd.) material and photos.

2 Selection of the Module

The Bluetooth serial module named even number is compatible with each other; The salve module is also compatible with each other. In other word, the function of HC-04 and HC-06, HC-03 and HC-05 are mutually compatible with each other. HC-04 and HC-06 are former version that user can't reset the work mode (master or slave). And only a few AT commands and functions can be used, like reset the name of Bluetooth (only the slaver), reset the password, reset the baud rate and check the version number. The command set of HC-03 and HC-05 are more flexible than HC-04 and HC-06's. Generally, the Bluetooth of HC-03/HC-05 is recommended for the user.

Here are the main factory parameters of HC-05 and HC-06. Pay attention to the differences:

HC-05	HC-06
Master and slave mode can be switched	Master and slave mode can't be switched
Bluetooth name: HC-05	Bluetooth name: linvor
Password:1234	Password:1234

Master role: have no function to remember the last paired salve device. It can be made paired to any slave device. In other words, just set AT+CMODE=1 when out of factory. If you want HC-05 to remember the last paired slave device address like HC-06, you can set AT+CMODE=0 after paired with the other device. Please refer the command set of HC-05 for the details.	Master role: have paired memory to remember last slave device and only make pair with that device unless KEY (PIN26) is triggered by high level. The default connected PIN26 is low level.
Pairing: The master device can not only make pair with the specified Bluetooth address, like cell-phone, computer adapter, slave device, but also can search and make pair with the slave device automatically. Typical method: On some specific conditions, master device and slave device can make pair with each other automatically. (This is the default method.)	Pairing: Master device search and make pair with the slave device automatically. Typical method: On some specific conditions, master and slave device can make pair with each other automatically.
Multi-device communication: There is only point to point communication for modules, but the adapter can communicate with multi-modules.	Multi-device communication: There is only point to point communication for modules, but the adapter can communicate with multi-modules.
AT Mode 1: After power on, it can enter the AT mode by triggering PIN34 with high level. Then the baud rate for setting AT command is equal to the baud rate in communication, for example: 9600. AT mode 2: First set the PIN34 as high level, or while on powering the module set the PIN34 to be high level, the Baud rate used here is 38400 bps. Notice: All AT commands can be operated only	AT Mode: Before paired, it is at the AT mode. After paired it's at transparent communication.

when the PIN34 is at high level. Only part of the AT commands can be used if PIN34 doesn't keep the high level after entering to the AT mode. Through this kind of designing, set permissions for the module is left to the user's external control circuit, that makes the application of HC-05 is very flexible.	
During the process of communication, the module can enter to AT mode by setting PIN34 to be high level. By releasing PIN34, the module can go back to communication mode in which user can inquire some information dynamically. For example, to inquire the pairing is finished or not.	During the communication mode, the module can't enter to the AT mode.
Default communication baud rate: 9600, 4800-1.3M are settable.	Default communication baud rate: 9600, 1200-1.3M are settable.
KEY: PIN34, for entering to the AT mode.	KEY: PIN26, for master abandons memory.
LED1: PIN31, indicator of Bluetooth mode. Slow flicker (1Hz) represents entering to the AT mode2, while fast flicker(2Hz) represents entering to the AT mode1 or during the communication pairing. Double flicker per second represents pairing is finished, the module is communicable. LED2: PIN32, before pairing is at low level, after the pairing is at high level. The using method of master and slaver's indicator is the same. Notice: The PIN of LED1 and LED2 are connected with LED+.	LED: The flicker frequency of slave device is 102ms. If master device already has the memory of slave device, the flicker frequency during the pairing is 110ms/s. If not, or master has emptied the memory, then the flicker frequency is 750m/s. After pairing, no matter it's a master or slave device, the LED PIN is at high level. Notice: The LED PIN connects to LED+ PIN.
Consumption: During the pairing, the current is	Consumption: During the pairing, the current is

fluctuant in the range of 30-40mA. The mean current is about 25mA. After paring, no matter processing communication or not, the current is 8mA. There is no sleep mode. This parameter is same for all the Bluetooth modules.	fluctuant in the range of 30-40 m. The mean current is about 25mA. After paring, no matter processing communication or not, the current is 8mA. There is no sleep mode. This parameter is same for all the Bluetooth modules.
Reset: PIN11, active if it's input low level. It can be suspended in using.	Reset: PIN11, active if it's input low level. It can be suspended in using.
Level: Civil	Level: Civil

The table above that includes main parameters of two serial modules is a reference for user selection.

HC-03/HC-05 serial product is recommended.

3. Information of Package

The PIN definitions of HC-03, HC-04, HC-05 and HC-06 are kind of different, but the package size is the same: 28mm * 15mm * 2.35mm.

The following figure 1 is a picture of HC-06 and its main PINs. Figure 2 is a picture of HC-05 and its main PINs. Figure 3 is a comparative picture with one coin. Figure 4 is their package size information. When user designs the circuit, you can visit the website of Guangzhou HC Information Technology Co., Ltd. (www.wavesen.com) to download the package library of protle version.

Figure 1 HC-06

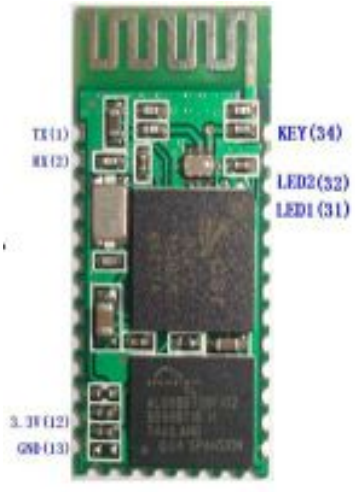


Figure 2 HC-05

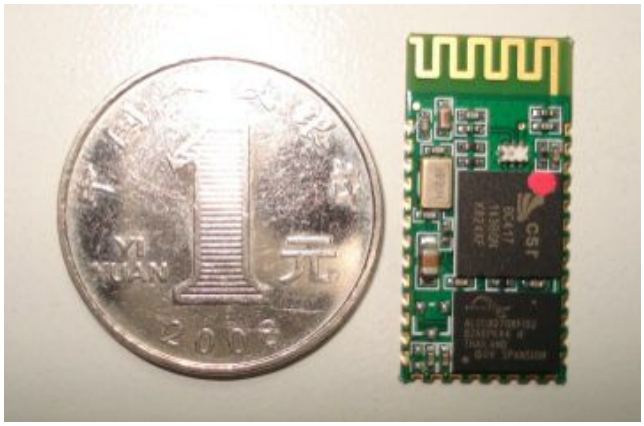
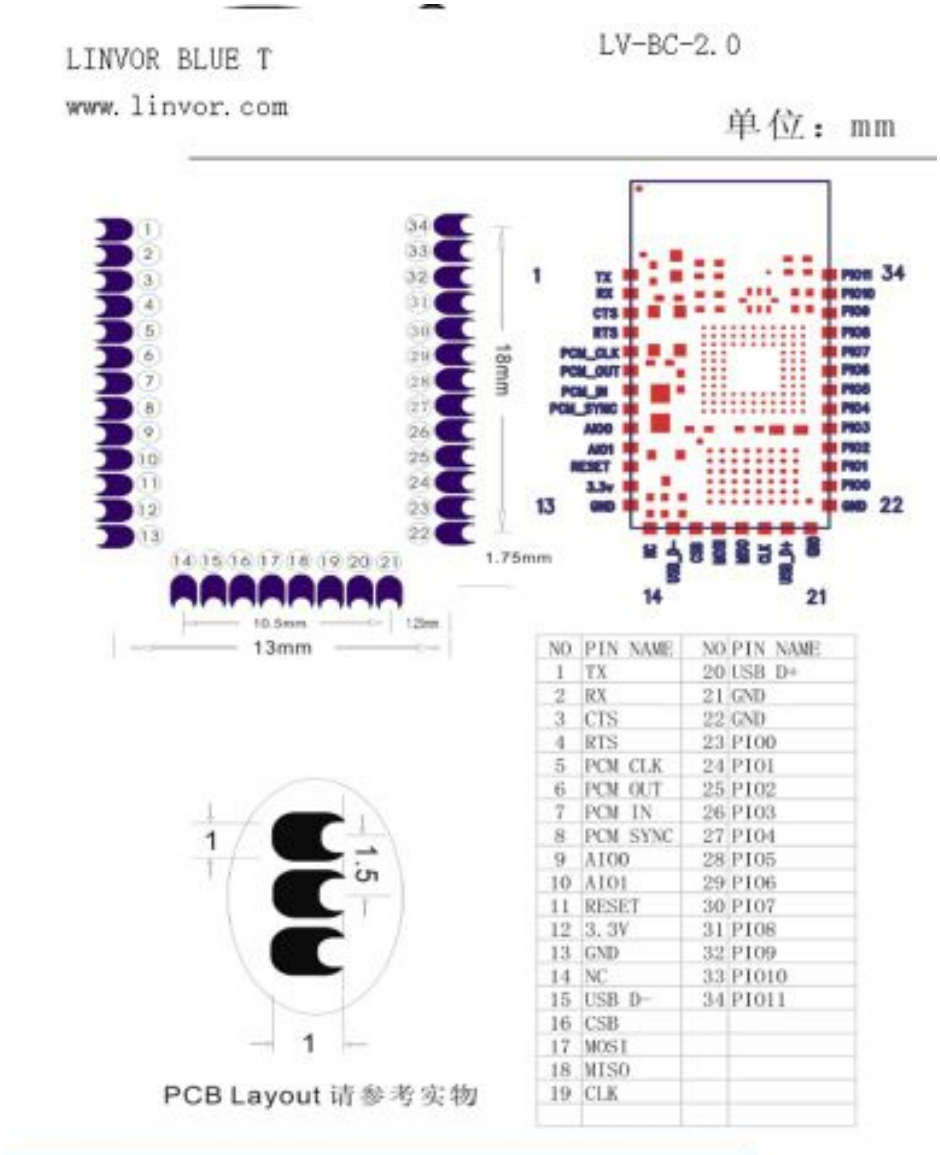


Figure 3 Comparative picture with one coin



NO	PIN NAME	NO	PIN NAME
1	TX	20	USB D+
2	RX	21	GND
3	CTS	22	GND
4	RTS	23	PIO0
5	PCM_CLK	24	PIO1
6	PCM_OUT	25	PIO2
7	PCM_IN	26	PIO3
8	PCM_SYNC	27	PIO4
9	AIO0	28	PIO5
10	AIO1	29	PIO6
11	RESET	30	PIO7
12	3.3V	31	PIO8
13	GND	32	PIO9
14	NC	33	PIO10
15	USB D-	34	PIO11
16	CSB		
17	MOSI		
18	MISO		
19	CLK		

Figure 4 Package size information

4. The Using and Testing Method of HC-06 for the First Time

This chapter will introduce the using method of HC-06 in detail. User can test the module according to this chapter when he or she uses the module at the first time.

PINs description:

PIN1	UART_TXD , TTL/CMOS level, UART Data output
PIN2	UART_RXD, TTL/COMS level, s UART Data input
PIN11	RESET, the reset PIN of module, inputting low level can reset the module, when the module is in using, this PIN can connect to air.
PIN12	VCC, voltage supply for logic, the standard voltage is 3.3V, and can work at 3.0-4.2V
PIN13	GND
PIN22	GND
PIN24	LED, working mode indicator Slave device: Before paired, this PIN outputs the period of 102ms square wave. After paired, this PIN outputs high level. Master device: On the condition of having no memory of pairing with a slave device, this PIN outputs the period of 110ms square wave. On the condition of having the memory of pairing with a slave device, this PIN outputs the period of 750ms square wave. After paired, this PIN outputs high level.
PIN26	For master device, this PIN is used for emptying information about pairing. After emptying, master device will search slaver randomly, then remember the address of the new got slave device. In the next power on, master device will only search this address.

(1) The circuit 1 (connect the module to 3.3V serial port of MCU) is showed by figure 5.

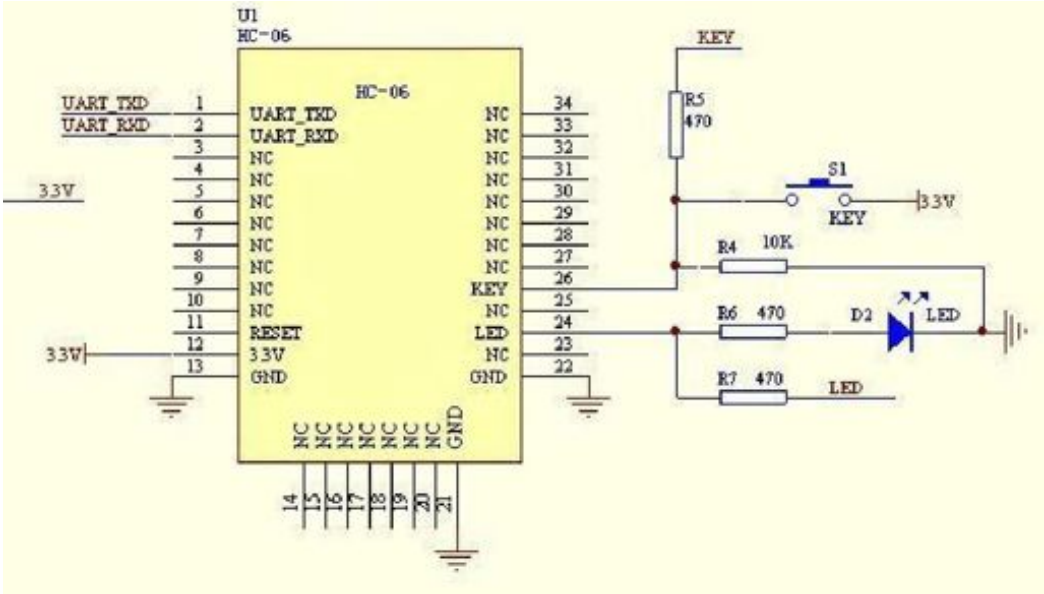


Figure 5 The circuit 1

In principle, HC-06 can work when UART_TXD, UART_RXD, VCC and GND are connected. However, for better testing results, connecting LED and KEY are recommended (when testing the master).

Where, the 3.3V TXD of MCU connects to HC-06's UART_RXD, the 3.3V RXD of MCU connects to HC-06's UART_TXD, and 3.3V power and GND should be connected. Then the minimum system is finished.

Note that, the PIN2:UART_RXD of Bluetooth module has no pull-up resistor. If the MCU TXD doesn't have pull-up function, then user should add a pull-up resistor to the UART_RXD. It may be easy to be ignored.

If there are two MCU which connect to master and slave device respectively, then before paired(LED will flicker) user can send AT commands by serial port when the system is power on. Please refer to HC-04 and HC-06's data sheet for detailed commands. In the last chapter, the command set will be introduced. Please pay attention to that the command of HC-04/HC-06 doesn't have terminator. For example, consider the call command, sending out AT is already enough, need not add the CRLF (carriage return line feed).

If the LED is constant lighting, it indicates the pairing is finished. The two MCUs can communicate with each other by serial port. User can think there is a serial port line between two MCUs.

(2) The circuit 2 (connect the module to 5V serial port of MCU) is showed by figure 6.

Figure 6 is the block diagram of Bluetooth baseboard. This kind of circuit can amplify Bluetooth module's operating voltage to 3.1-6.5V. In this diagram, the J1 port can not only be connected with MCU system of 3.3V and 5V, but also can be connected with computer serial port.

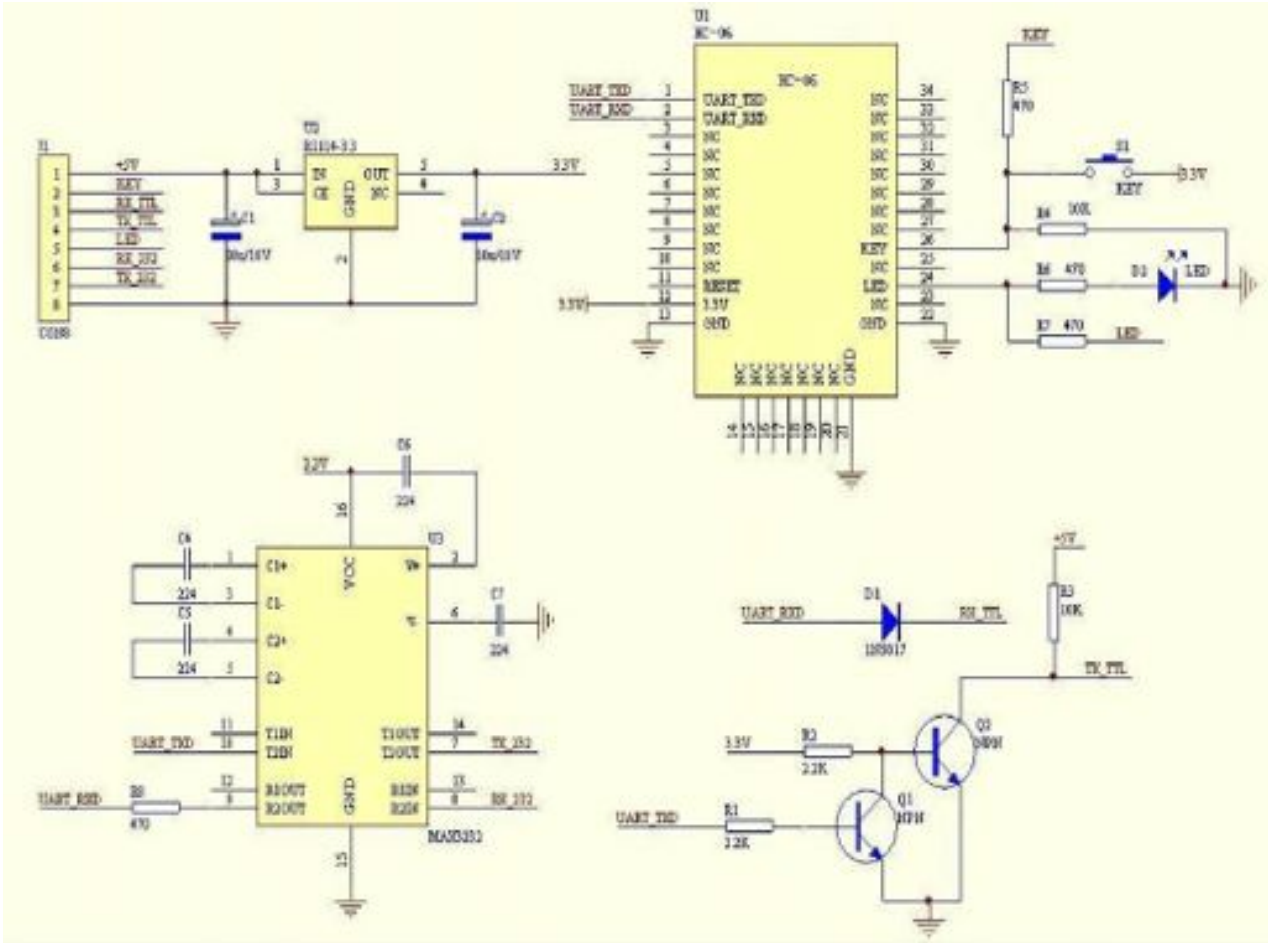


Figure 6 The circuit 2

(3) AT command test

Before paired, the mode of HC-04 and HC-06 are AT mode.

On the condition of 9600N81, OK will be received when user send the two letters AT. Please refer to the last chapter of datasheet for other commands of HC-06. Please pay attention to that sending out AT is already enough, need not add the CRLF (carriage return line feed).

The command set of Version V1.4 doesn't include parity. The version V1.5 and its later version have parity function. Moreover, there are three more commands of V1.5 than V1.4. They are:

No parity (default) AT+PN

Odd parity	AT+PO
Even parity	AT+PE

Do not let the sending frequency of AT command of HC-06 exceed 1Hz, because the command of HC-06 end or not is determined by the time interval.

(4) Pairing with adapter

User can refer to the download center of the company's website for "The Introduction of IVT" that introduces the Bluetooth module makes pair with computer adapter. That document taking HC-06-D for example introduces how the serial module makes pair with the adapter. That method is like to make pair with cell-phone. But the difference is that cell-phone need a third-party communication software to help. It's liked the kind of PC serial helper of and the hyper terminal. A software named "PDA serial helper" provided by our company is suitable for WM system. It has been proven that this serial module is supported by many smart phone systems' Bluetooth, such as, sybian, android, windows mobile and etc.

(5) Pairing introduction

HC-06 master device has no memory before the first use. If the password is correct, the mater device will make pair with the slave device automatically in the first use. In the following use, the master device will remember the Bluetooth address of the last paired device and search it. The searching won't stop until the device is found. If master device's PIN26 is input high level, the device will lose the memory. In that occasion, it'll search the proper slave device like the first use. Based on this function, the master device can be set to make pair with the specified address or any address by user.

(6) Reset new password introduction

User can set a new password for the HC-06 through AT+PINxxxx command. But the new password will become active after discharged all the energy of the module. If the module still has any energy, the old one is still active. In the test, for discharging all the system energy and activating the new password, we can connect the power supply PIN with GND about 20 seconds after the power is cut off. Generally, shutting down the device for 30 minutes also can discharge the energy, if there is no peripheral circuit helps discharge energy. User should make the proper way according to the specific situation.

(7) Name introduction

If the device has no name, it's better that user doesn't try to change the master device name. The name should be limited in 20 characters.

Summary: The character of HC-06: 1 not many command 2 easy for application 3 low price. It's good for some specific application. HC-04 is very similar with HC-06. Their only one difference is HC-04 is for industry, HC-06 is for civil. Except this, they don't have difference.

The following reference about HC-04 and HC-06 can be downloaded from company website www.wavesen.com:

HC-06 datasheet .pdf	(the command set introduction is included)
HC-04 datasheet .pdf	(the command set introduction is included)
IVT BlueSoleil-2.6	(IVT Bluetooth drive test version)
Bluetooth FAQ.pdf	
HC-04-D(HD-06-D)datasheet(English).pdf	
HC-06-AT command software (test version)	(some commands in V1.5 is not supported by V1.4)
PCB package of Bluetooth key modules	(PCB package lib in protel)
IVT software manual.pdf	(introduce how to operate the modern and make pair with Bluetooth module)
PDA serial test helper.exe	(serial helper used for WM system)

5 manual for the first use of HC-05

This chapter will introduce how to test and use the HC-05 if it's the first time for user to operate it.

(1) PINs description

PIN1	UART_TXD, Bluetooth serial signal sending PIN, can connect with MCU's RXD PIN
PIN2	UART_RXD, Bluetooth serial signal receiving PIN, can connect with the MCU's TXD PIN, there is no pull-up resistor in this PIN. But It needs to be added an eternal pull-up resistor.
PIN11	RESET, the reset PIN of module, inputting low level can reset the module, when the module is in using, this PIN can connect to air.
PIN12	VCC, voltage supply for logic, the standard voltage is 3.3V, and can work at 3.0-4.2V
PIN13	GND

PIN31	LED1, indicator of work mode. Has 3 modes: When the module is supplied power and PIN34 is input high level, PIN31 output 1Hz square wave to make the LED flicker slowly. It indicates that the module is at the AT mode, and the baud rate is 38400; When the module is supplied power and PIN34 is input low level, PIN31 output 2Hz square wave to make the LED flicker quickly. It indicates the module is at the pairable mode. If PIN34 is input high level, then the module will enter to AT mode, but the output of PIN31 is still 2Hz square wave. After the pairing, PIN31 output 2Hz square ware. Note: if PIN34 keep high level, all the commands in the AT command set can be in application. Otherwise, if just excite PIN34 with high level but not keep, only some command can be used. More information has provided at chapter 2.
PIN32	Output terminal. Before paired, it output low level. Once the pair is finished, it output high level.
PIN34	Mode switch input. If it is input low level, the module is at paired or communication mode. If it's input high level, the module will enter to AT mode. Even though the module is at communication, the module can enter to the AT mode if PIN34 is input high level. Then it will go back to the communication mode if PIN34 is input low level again.

(2) Application circuit 1 (connect to the 3.3V system)

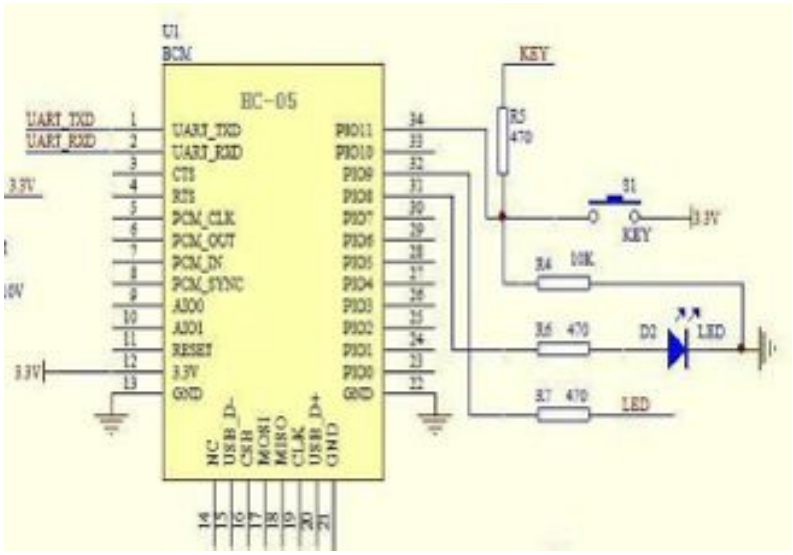


Figure 7 Application 1

(3) Application circuit 2 (connect to 5V serial system or PC serial)

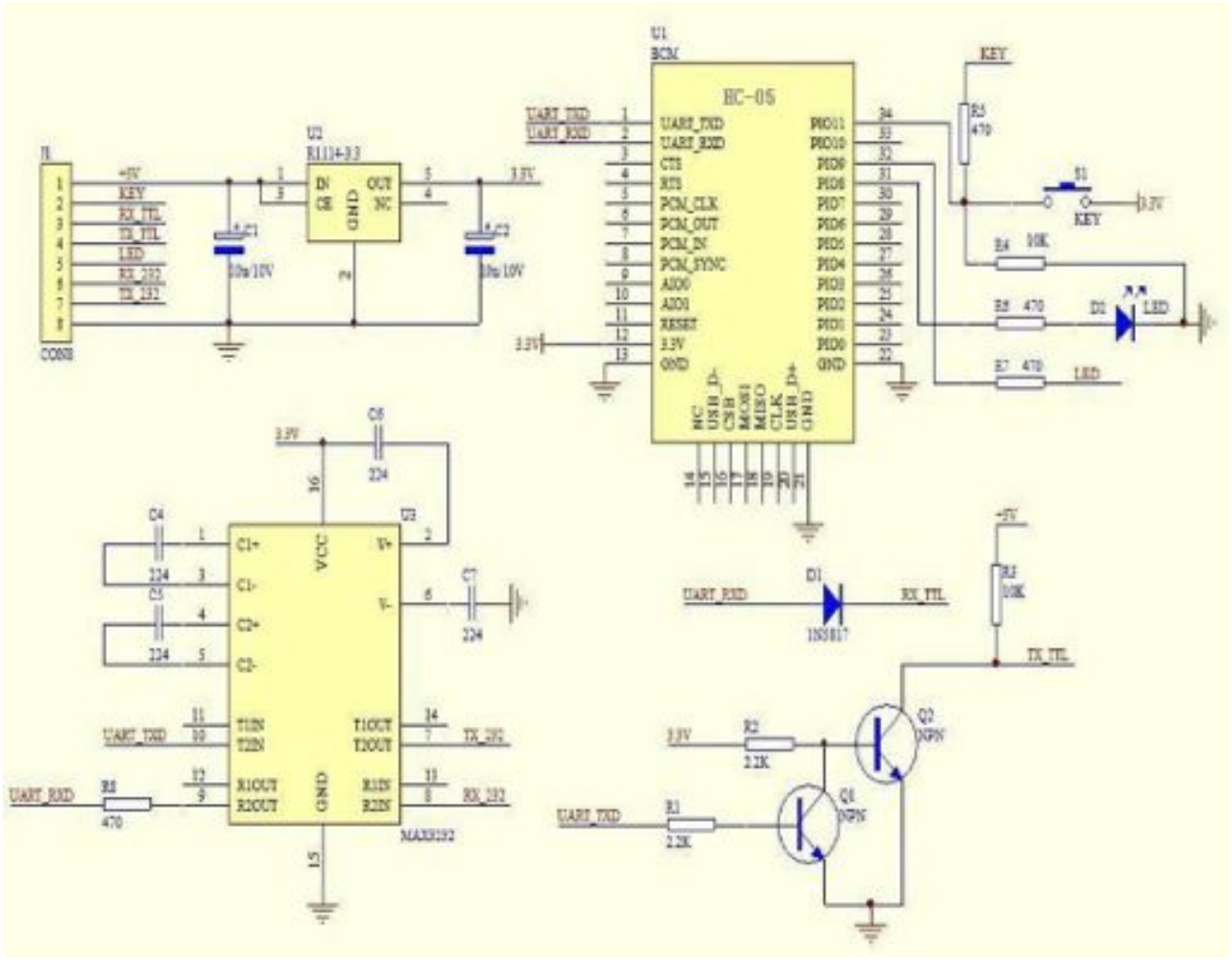


Figure 8 Application circuit 2

(4) AT command test

This chapter introduces some common commands in use. The detail introduction about HC-05 command is in HC-0305 AT command set.

Enter to AT mode:

Way1: Supply power to module and input high level to PIN34 at the same time, the module will enter to AT mode with the baud rate-38400.

Way2: In the first step, supply power to module; In the second step, input high level to PIN34. Then the module will enter to AT mode with the baud rate-9600. Way1 is recommended.

Command structure: all command should end up with "\r\n" (Hex: 0X0D X0A) as the terminator. If

the serial helper is installed, user just need enter “ENTER” key at the end of command.

Reset the master-slave role command:

AT+ROLE=0 ----Set the module to be salve mode. The default mode is salve.

AT+ROLE=1 ----Set the module to be master mode.

Set memory command:

AT+CMODE=1

Set the module to make pair with the other random Bluetooth module (Not specified address). The default is this mode.

AT+CMODE=1

Set the module to make pair with the other Bluetooth module (specified address). If set the module to make pair with random one first, then set the module to make pair with the Bluetooth module has specified address. Then the module will search the last paired module until the module is found.

Reset the password command

AT+PSWD=XXXX

Set the module pair password. The password must be 4-bits.

Reset the baud rate

AT+UART== <Param>,<Param2>,<Param3>.

More information is provided at HC-0305 command set

Example:

AT+UART=9600,0,0 ----set the baud rate to be 9600N81

Reset the Bluetooth name

AT+NAME=XXXXX

Summary:

HC-05 has many functions and covers all functions of HC-06. The above commands are the most common ones. Besides this, HC-05 leaves lots of space for user. So HC-05 is better than HC-06 and

recommended. HC-03 is similar with HC-05. The above introduction also suits HC-03

The following reference about HC-03 and HC-05 can be downloaded from company website www.wavesen.com:

HC-03 datasheet .pdf	(the command set introduction is included)
HC-05 datasheet .pdf	(the command set introduction is included)
IVT BlueSoleil-2.6	(IVT Bluetooth drive test version)
Bluetooth FAQ.pdf	
PCB package of Bluetooth key modules	(PCB package lib in protel)
IVT software manual.pdf	(introduce how to operate the modern and make pair with Bluetooth module)
PDA serial test helper.exe	(serial helper used for WM system)
HC-03/05 Bluetooth serial command set.pdf	

6. Ordering information

The website of Guangzhou HC Information Technology Co., Ltd is www.wavesen.com The contact information is provided at the company website.

Order Way: If you want our product, you can give order to the production center of our company directly or order it in Taobao. There is a link to Taobao in our company website.

Package: 50 pieces chips in an anti-static blister package. The weight of a module is about 0.9g. The weight of a package is about 50g.

Please provide the product's model when you order:

HC-04-M HC-04 master module

HC-04-S HC-04 slave module

HC-06-M HC-06 master module

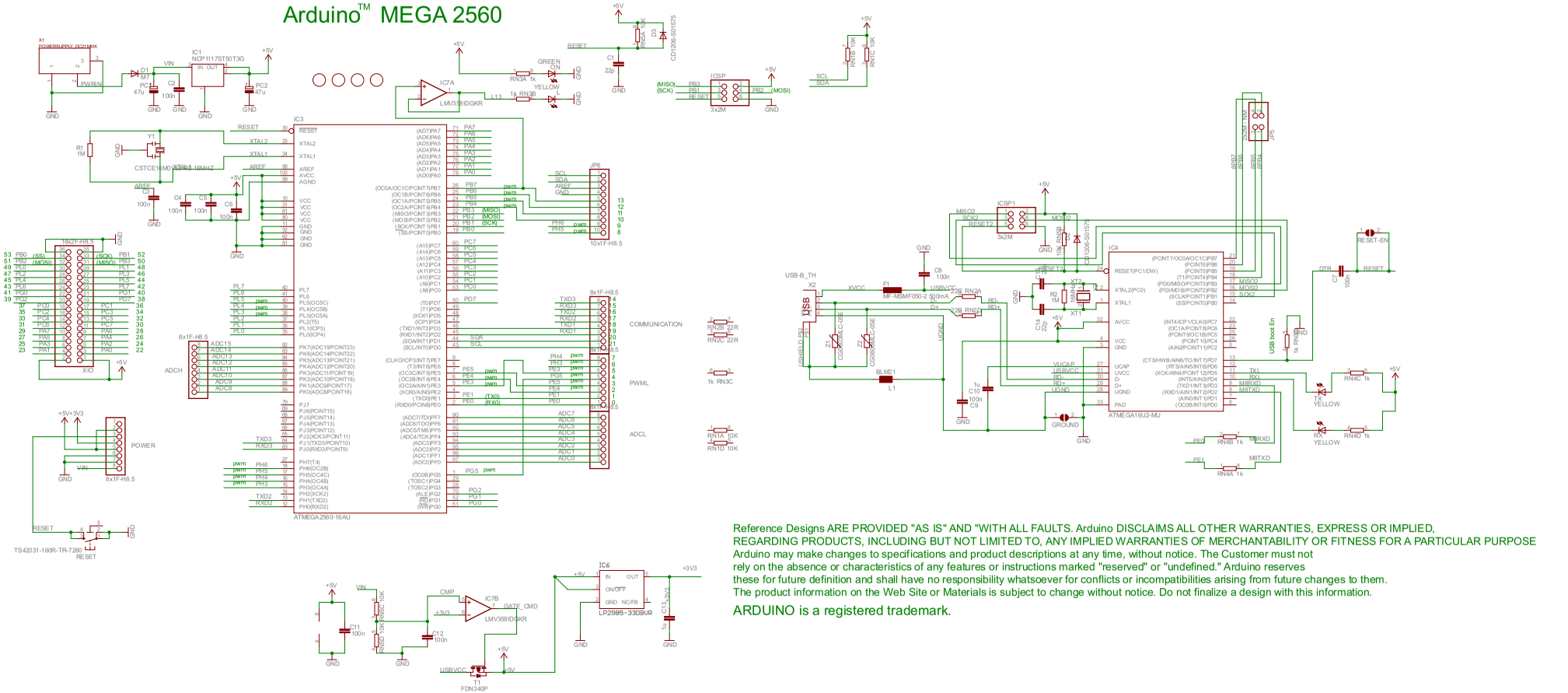
HC-06-S HC-06 slave module

HC-03

HC-05 HC-03/05 can be preset to be master module or slave module.

ANEXO D - Esquemático del Arduino Mega 2560

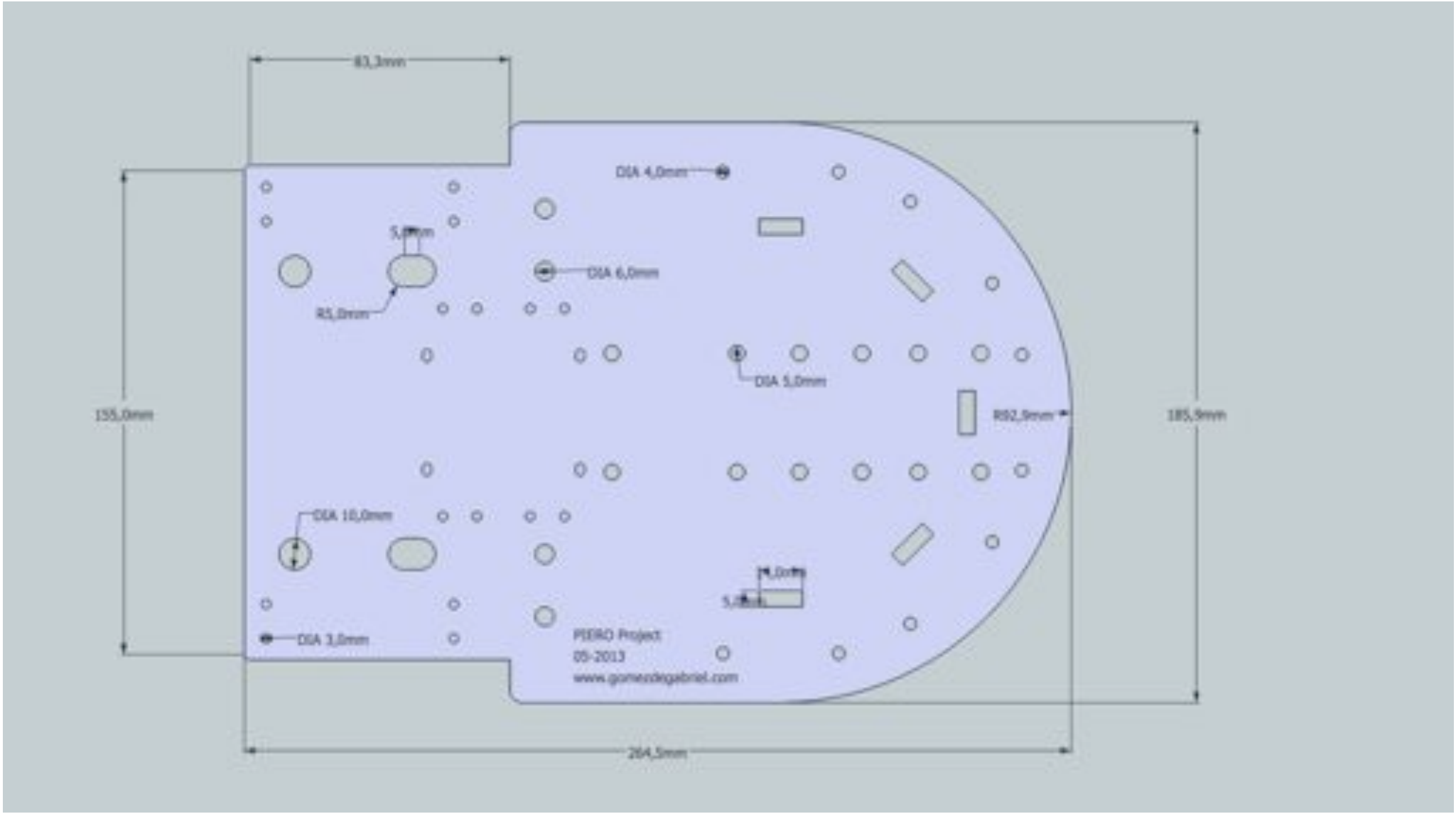
Arduino™ MEGA 2560



Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

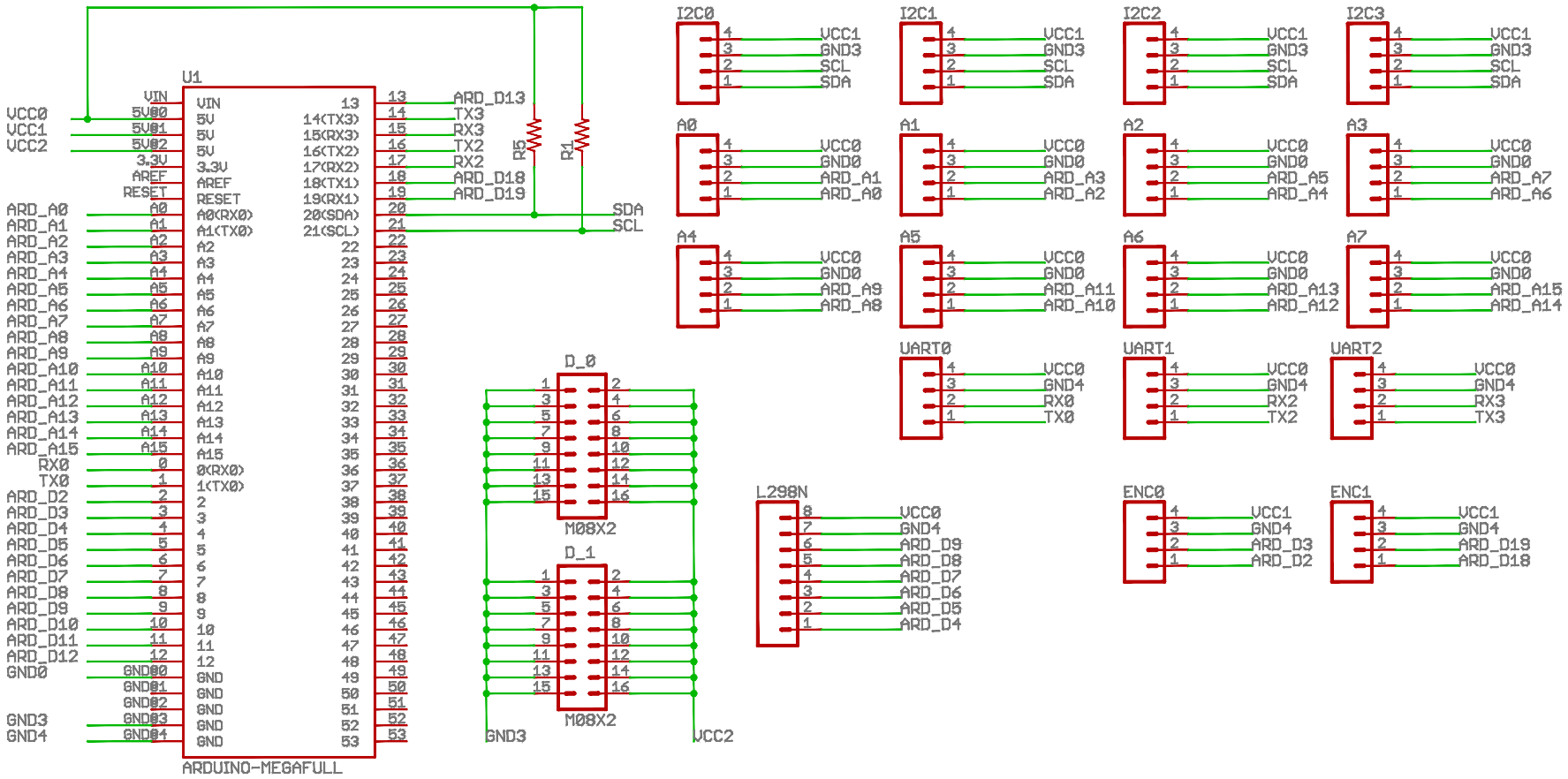
ARDUINO is a registered trademark.

ANEXO E - Esquema de la Plataforma Base


83,3mm
DIA 4,0mm
5,0mm
DIA 6,0mm
R5,0mm
DIA 5,0mm
155,0mm
R82,9mm
185,9mm
DIA 10,0mm
DIA 3,0mm
PEIRO Project
05-2013
www.gomezdegabriel.com
264,5mm

IV

PLANOS Y ESQUEMAS

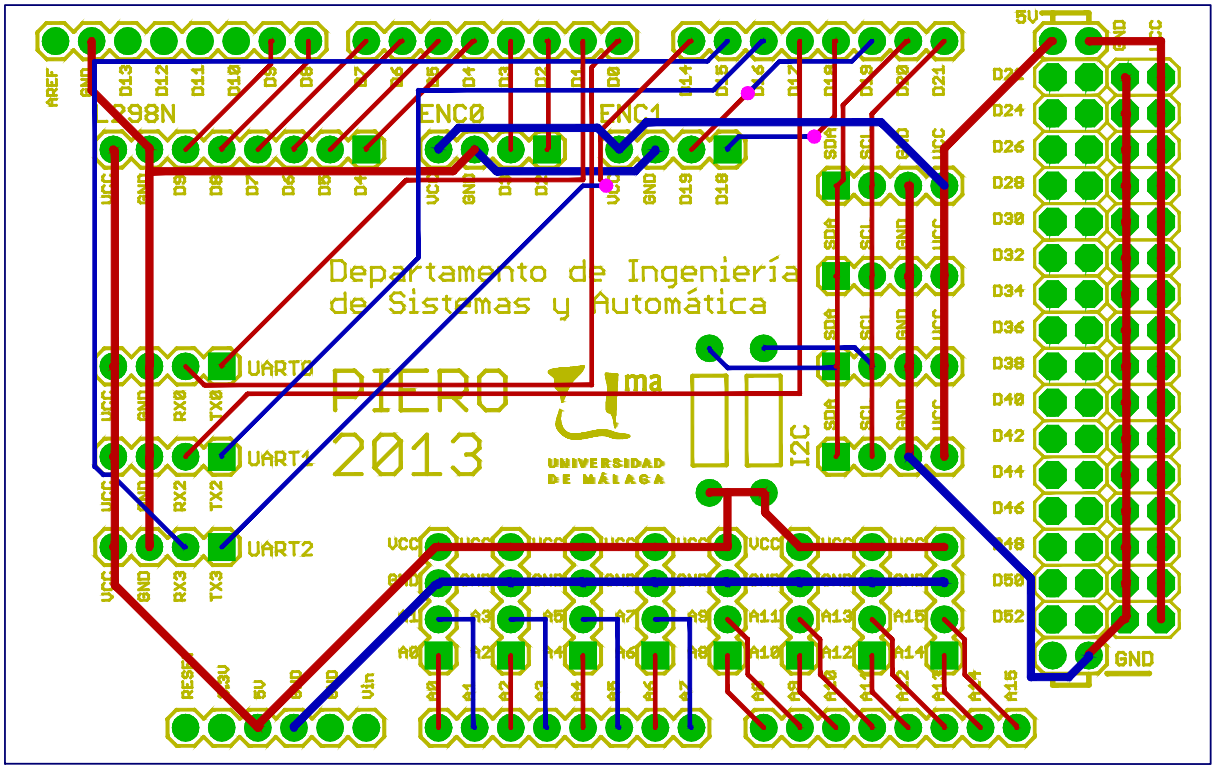


Universidad de Málaga

E.T.S. de Ingeniería Industrial

"Sistema Didáctico para Programación de Robots Móviles en Matlab/Simulink"

Autor: Juan Enrique Gil Lozano Fdo:	Esquemático PieroShield	Málaga a 4 de Junio de 2014 ESCALA Sin Escala PLANO Nº 1



LEYENDA	
	Líneas de la capa superior.
	Líneas de la capa inferior.
	Pads.
	Serigrafía.
	Vías.

Universidad de Málaga

E.T.S. de Ingeniería Industrial

"Sistema didáctico para programación de robotsmóviles en Matlab/Simulink"

Autor: Juan Enrique Gil Lozano

Fdo:

Diseño PieroShield (Serigrafía, Capa Sup. y Capa Inf.)

Málaga a 4 de Junio de 2014

ESCALA 2:1

PLANO Nº 2